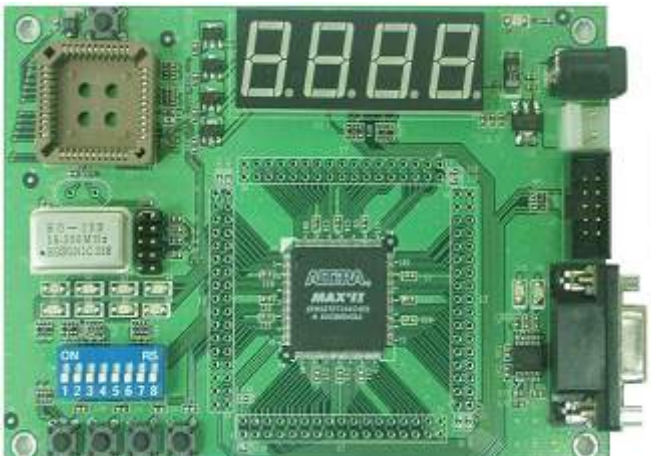
GFEC MAX II Starter Kit

使用說明書



茂綸股份有限公司
GALAXY FAR EAST CORP
www.gfec.com.tw

Address : 14F , 207-5, Sec 3,Peihsin Rd.,
Hsintien, Taiwan R.O.C.
TEL: 886-2-8913-2200 FAX: 886-2-8913-2277

聯絡茂綸股份有限公司

台北總公司
地址:<231>台北縣新店市北新路三段207-5號14F
TEL:886-2-8913-2200
FAX:886-2-8913-2277

新竹分公司
地址: <300>新竹市光復路一段526號3F
TEL:886-3-578-6766
FAX:886-3-577-4795

高雄分公司
地址:<800>高雄市新興區民生一路56號10F之8
TEL:886-7-223-1338
FAX:886-7-222-4051

技術支援專線
TEL:0800819595

茂綸全球資訊網頁
HTTP://WWW.GFEC.COM.TW

茂綸股份有限公司
GALAXY FAR EAST CORP
www.gfec.com.tw

目錄

第一章　使用簡介.....4

第二章　Quartus II 快速入門.....18

第三章　測試程式下載說明.....35

第四章　ALU 範例.....46

第五章　七段顯示器範例.....49

第六章　時鐘範例.....52

第七章　紅綠燈範例.....60

第八章　遊戲機範例.....67

第九章　RS-232 範例.....76

附錄 A　QuartusII 4.2 安裝流程.....90

附錄 B　License 申請流程.....95

附錄 C　Windows XP/2000 下安裝 Download cable.....100

附錄 D　MAX II Starter Kit Schematic.....114

第一章　使用簡介

Contents

- Altera MAX II Board
- Download Cable
- AC Adapter (Input：AC 115 V 60Hz / Output：DC：6V 1A) or USB to DC Cable (僅提供一種)
- CD-ROM

Attention

- 當您拿到本實驗板後，請確認盒內東西是否完整
- 本實驗板擴充 I/O 電壓最多只能接受 3.3V 的信號，若不甚燒毀 IC (EPM1270T144C5ES)，需向本公司購買此 IC 並自行焊接處理
- 在正常使用下，本實驗板提供 3 個月的保固期

Feature

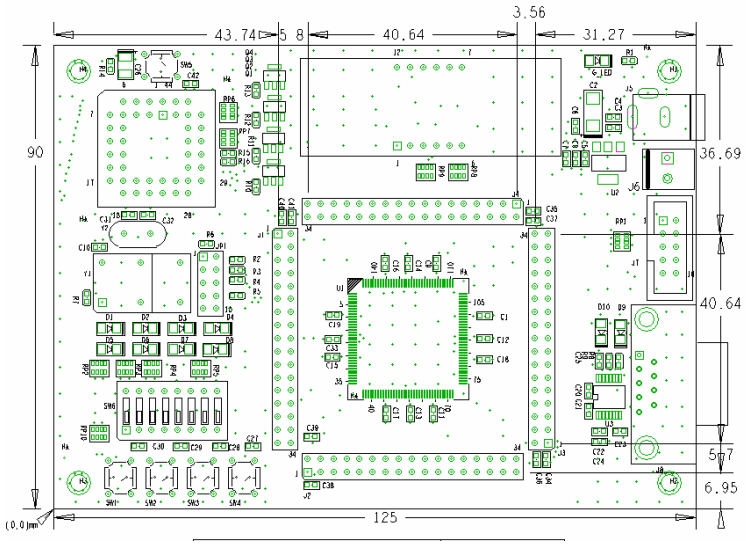
本實驗板是專門為 Altera MAX II Device Family 訂製的模擬板，若針對數位設計有興趣者可利用此實驗板模擬與實現自己想要的東西，本實驗板內建的 CPLD 為 EPM1270T144C5ES，此 IC 提供 1270 LEs，116 支一般 I/0，8K bits User Flash Memory。詳細的資料與規格可由本實驗板提供之光碟片或至 Altera Web Sit 取得 (http://www.altera.com)。

除了 EPM1270T144C5ES 這顆 IC 之外，本實驗板還有提供一些週邊零件連接至 EPM1270T144C5ES 供大家使用，使用者可配合這些零件快速的完成一個數位邏輯設計。

週邊零件

- 4 Digitals seven segment LED Displayer
- 8 Bits DIP Switch
- 4 Push Buttons
- 8 LEDs
- 1 16Mhz Oscillator
- 1 UART Connector
- 4 Extension I/O Connectors
- 44 Pin Socket (for 3.3V 8051 CPU)

MAX II 實驗板外觀圖



以下是針對本實驗板周邊配備之詳細資料

4 Digitals seven segment LED Displayer (DS1)

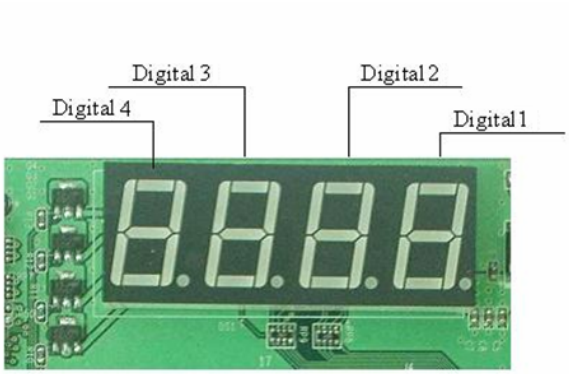
LED Displayer 是由 4 組 7 段顯示器所組成，七段顯示器的每一段 LED 相互連接在一起，另外還各有一支 common pin 連接至其中一組七段顯示器。當其中一支 common pin 為高準位時而且 7 段顯示器中的其中一段 LED 為低準位時，此七段

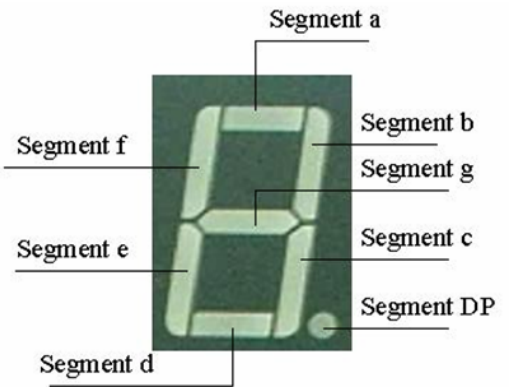
顯示器之其中一段 LED 將亮起。雖然 4 組 7 段顯示器中的每一段 LED 是連接在一起的，但我們可以利用 common pin 來決定到底啟動哪一組 7 段顯示器。一般來說我們都是利用掃描 common pin 的方式來呈現 4 個不同的數字。

裝置	4 Digitals seven segment LED Displayer							
Segment pin 代號	a	b	c	d	e	f	g	DP
MAX II 腳位	37	38	39	40	41	42	43	44

裝置	4 Digitals seven segment LED Displayer			
Common pin 代號	Digital 1	Digital 2	Digital 3	Digital 4
MAX II 腳位	29	30	31	32

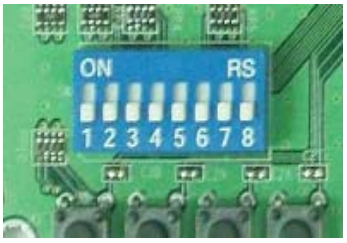
備註：當 Common pin 為高準位時，表示啟動其中一組七段顯示器
當 Segment pin 為低準位時，表示點亮那一段 LED





8 Bits DIP Switch (SW6)

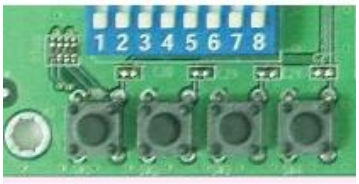
本實驗板提供 8 bits DIP Switch，當 Switch 切至"ON"時，MAX II Device I/O 將為 Low，其中 8 bits DIP Switch 的信號接腳和 MAX II Device 的 I/O 關係可參考下表。



裝置	8 Bits DIP Switch							
Switch 代號	DIP1	DIP2	DIP3	DIP 4	DIP 5	DIP 6	DIP 7	DIP 8
MAX II 腳位	22	21	16	15	14	13	12	11

備註：ON 為低準位，OFF 為高準位

4 Push Button (SW1 – SW4)



本實驗板提供 4 個 Push button 當按下此按鍵時，將提供 Low 的信號給 MAX II，不按時，此接腳將 Pull-High，其中 Push Button 的信號接腳和 MAX II Device 的 I/O 關係可參考下表。

裝置	4 Push Button			
Push Button 代號	SW1	SW2	SW3	SW 4
MAX II 腳位	23	24	27	28

備註：按下時為低準位，不按時為高準位

8 LEDs (D1 – D8)

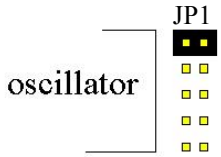
當 MAX II I/O 送出 Low 的信號時，LED 將亮起，其中 8 個 LED 的信號接腳和 MAX II Device 的 I/O 關係可參考下表。

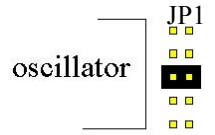
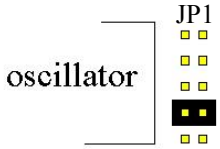
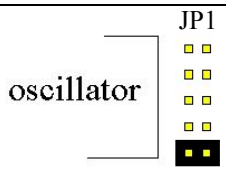
裝置	8 LEDs							
LED 代號	D1	D2	D3	D 4	D 5	D 6	D 7	D 8
MAX II 腳位	1	2	3	4	5	6	7	8

備註：低準位 LED 亮，高準位 LED 不亮

16M Hz Oscillator

本實驗板提供 1 個 16M Hz 全長型的振盪器，此震盪器可經由 JP1 並利用 Jumper 將 clock 送至 MAX II Device，其中 16M Hz 的全長型的振盪器及 JP1 的信號接腳和 MAX II Device 的 I/O 關係可參考下表。

裝置 JP1	MAX II 腳位
	18 / GCLK0P
	20 / GCLK1P

	89 / GCLK2P
	91 / GCLK3P
	44 Pin PLCC Socket (8051 MPU) Pin21

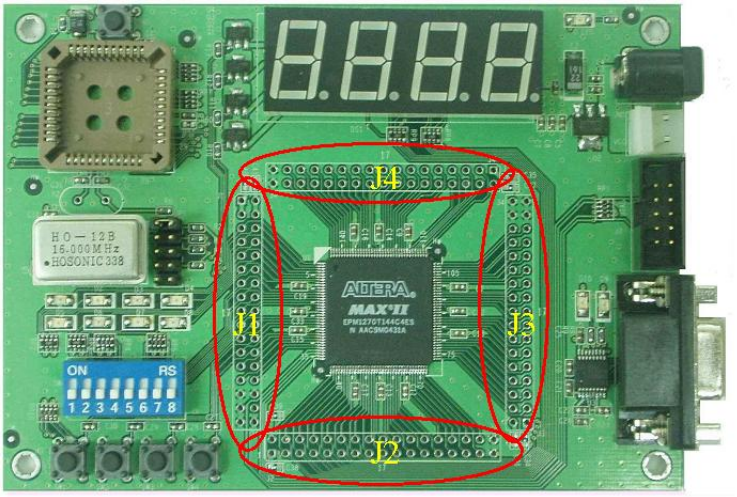
備註：JP1 可同時插入多個 Jumper 來提供多組 clk 輸入

UART Connector (J8)

本實驗板提供一個 UART 的 connector，使用者可自行撰寫一個 UART 的 controller 在 MAX II Device 裡，則可利用它和其它擁有 UART 裝置的介面做溝通。其中 UART 電路是經由 3 個部分所組成，第一部分是 UART 信號的轉換(Intersil / ICL3232CA)，它能將 MAX II I/O 3.3 V 的電壓準位和 RS-232 的電壓準位(+ / - 12V)做轉換，第二部分是兩個 LED (D9,D10)，可利用 D9（傳送），D10（接收）來判斷 UART 是在傳送還是接收資料。第三部分為 RS-232 母座連接器，可利用標準之 RS-232 Cable 和 PC 連接，其中 UART Connector 和 MAX II Device 的 I/O 關係可參考下表。

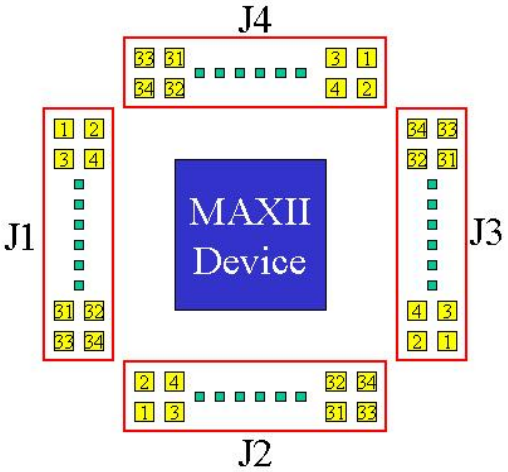
裝置	UART Connector			
UART 代號	TXD	CTS	RXD	RTS
Connector (J8)腳位	2	8	3	7
MAX II 腳位	45	48	49	50

4 Extension I/O Connectors (J1 / J2 / J3/ J4)



EPM1270T144C5ES 提供 116 支一般 I/O pin，本實驗板擴充 I/O 部分分別連接 J1，J2，J3，J4 (2 X 17 connector)，如此能讓使用人員很容易的利用此連接器連接至其他電路板，J1 to J4 也提供了 Power Pin 和 GND Pin，Power Pin 提供 3.3V 的電壓，但是沒有過電流保護措施，假設您的擴充電路板需要從本實驗板提供大電流時(300 mA 以上)，建議使用者在自己的擴充電路板準備好電源電路，再將兩邊的地連接在一起。除了 Power Pin 和 GND Pin 外，J1 to J4 連接器上的其它 Pin 直接連接至 MAX II Device I/O 上，所以禁止將超過 3.3V 的信號連接至此連接器。

除此之外，部分 Pin 腳除了連接至 MAX II Device 外,也有連接至實驗板上的其它零件(例如：LED，Push Button 等等)，故使用者也要注意這些零件是否會影響到擴充 I/O 接腳。其中這些擴充 I/O 連接器和 MAX II Device 的 I/O 關係可參考以下表格



MAX II 腳位	Connector	特性	MAX II 腳位	Connector	特性
	J1 Pin1	VCC / 3.3V		J2 Pin1	VCC / 3.3V
	J1 Pin2	VCC / 3.3V		J2 Pin2	VCC / 3.3V
	J1 Pin3	VCC / 3.3V	37	J2 Pin3	Segment a
	J1 Pin4	VCC / 3.3V	38	J2 Pin4	Segment b
1	J1 Pin5	LED D1	39	J2 Pin5	Segment c
2	J1 Pin6	LED D2	40	J2 Pin6	Segment d
3	J1 Pin7	LED D3	41	J2 Pin7	Segment e
4	J1 Pin8	LED D4	42	J2 Pin8	Segment f
5	J1 Pin9	LED D5	43	J2 Pin9	Segment g
6	J1 Pin10	LED D6	44	J2 Pin10	Segment DP
7	J1 Pin11	LED D7	45	J2 Pin11	RS232 TXD
8	J1 Pin12	LED D8	48	J2 Pin12	RS232 DTS
11	J1 Pin13	DIP8	49	J2 Pin13	RS232 RXD
12	J1 Pin14	DIP7	50	J2 Pin14	RS232 RTS
13	J1 Pin15	DIP6	51	J2 Pin15	U4 PIN2 / P1.0
14	J1 Pin16	DIP5	52	J2 Pin16	U4 PIN3 / P1.1
15	J1 Pin17	DIP4	53	J2 Pin17	U4 PIN4 / P1.2
16	J1 Pin18	DIP3	55	J2 Pin18	U4 PIN5 / P1.3
21	J1 Pin19	DIP2	57	J2 Pin19	U4 PIN6 / P1.4
22	J1 Pin20	DIP1	58	J2 Pin20	U4 PIN7 / P1.5
23	J1 Pin21	SW1	59	J2 Pin21	U4 PIN8 / P1.6
24	J1 Pin22	SW2	60	J2 Pin22	U4 PIN9 / P1.7
27	J1 Pin23	SW3	61	J2 Pin23	U4 PIN10 / RST
28	J1 Pin24	SW4	62	J2 Pin24	U4 PIN11 / P3.0(RXD)
29	J1 Pin25	Digital 1	63	J2 Pin25	U4 PIN13 / P3.1(TXD)
30	J1 Pin26	Digital 2	66	J2 Pin26	U4 PIN14 / P3.2(INT0)
31	J1 Pin27	Digital 3	67	J2 Pin27	U4 PIN15 / P3.3(INT1)
32	J1 Pin28	Digital 4	68	J2 Pin28	U4 PIN16 / P3.4(T0)
	J1 Pin29	GND	69	J2 Pin29	U4 PIN17 / P3.5(T1)
	J1 Pin30	GND	70	J2 Pin30	U4 PIN18 / P3.6
	J1 Pin31	GND	71	J2 Pin31	U4 PIN19 / P3.7
	J1 Pin32	GND	72	J2 Pin32	U4 PIN33 / ALE
	J1 Pin33	GND		J2 Pin33	GND
	J1 Pin34	GND		J2 Pin34	GND

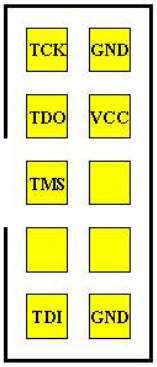
MAX II 腳位	Connector	特性	MAX II 腳位	Connector	特性
	J3 Pin1	VCC / 3.3V		J4 Pin1	VCC / 3.3V
	J3 Pin2	VCC / 3.3V		J4 Pin2	VCC / 3.3V
73	J3 Pin3	U4 PIN32 / PSEN	109	J4 Pin3	
74	J3 Pin4	U4 PIN31 / P2.7	110	J4 Pin4	
75	J3 Pin5	U4 PIN30 / P2.6	111	J4 Pin5	
76	J3 Pin6	U4 PIN29 / P2.5	112	J4 Pin6	
77	J3 Pin7	U4 PIN28 / P2.4	113	J4 Pin7	
78	J3 Pin8	U4 PIN27 / P2.3	114	J4 Pin8	
79	J3 Pin9	U4 PIN26 / P2.2	117	J4 Pin9	
80	J3 Pin10	U4 PIN25 / P2.1	118	J4 Pin10	
81	J3 Pin11	U4 PIN24 / P2.0	119	J4 Pin11	
84	J3 Pin12	U4 PIN43 / P0.0	120	J4 Pin12	
85	J3 Pin13	U4 PIN42 / P0.1	121	J4 Pin13	
86	J3 Pin14	U4 PIN41 / P0.2	122	J4 Pin14	
87	J3 Pin15	U4 PIN40 / P0.3	123	J4 Pin15	
88	J3 Pin16	U4 PIN39 / P0.4	124	J4 Pin16	
93	J3 Pin17	U4 PIN38 / P0.5	125	J4 Pin17	
94	J3 Pin18	U4 PIN37 / P0.6	127	J4 Pin18	
95	J3 Pin19	U4 PIN35 / R0.7	129	J4 Pin19	
96	J3 Pin20		130	J4 Pin20	
97	J3 Pin21		131	J4 Pin21	
98	J3 Pin22		132	J4 Pin22	
101	J3 Pin23		133	J4 Pin23	
102	J3 Pin24		134	J4 Pin24	
103	J3 Pin25		137	J4 Pin25	
104	J3 Pin26		138	J4 Pin26	
105	J3 Pin27		139	J4 Pin27	
106	J3 Pin28		140	J4 Pin28	
107	J3 Pin29		141	J4 Pin29	
108	J3 Pin30		142	J4 Pin30	
	J3 Pin31	GND	143	J4 Pin31	
	J3 Pin32	GND	144	J4 Pin32	
	J3 Pin33	GND		J4 Pin33	GND
	J3 Pin34	GND		J4 Pin34	GND

DC Input Connector

本本實驗板已經內建 3.3V/500mA 的電壓調節器，電源部分可利用本實驗板所提供的 AC Adapter 或 USB to DC Cable 直接插入 J5 即可(注意：AC Adapter 僅限用於 110V/60Hz 之電源)，也可利用電源供應器將電源連接至 J6 即可(建議輸入電壓 5～6V)，當您有電源輸入時，G_LED 將同時亮起。

JTAG 5X2 Header Connector

Altera MAX II Device 有提供 ISP (In System Programming) 功能，您可利用本實驗板所提供之 Download Cable，一邊插至電腦的並列埠，另一邊插至 J7，再利用 Altera 專用軟體 Quartus II，即可將資料下載至本實驗板。



44 Pin PLCC Socket for MPU(8051)

我們可將標準 3.3V 8051 MPU 放進此 Socket，本實驗板 8051 I/O Port 直接連接至 MAX II Device，也有提供 8051 CLK 和 Reset 按鈕，有關此 Socket 的定義可參考下表。

44 Pin PLCC	8051 MPU	MAX II I/O	44 Pin PLCC	8051 MPU	MAX II I/O
1	NC		23	NC	
2	P1.0 (T2)	51	24	P2.0(A8)	81
3	P1.1(T2 EX)	52	25	P2.1(A9)	80
4	P1.2	53	26	P2.2(A10)	79
5	P1.3	55	27	P2.3(A11)	78
6	P1.4	57	28	P2.4(A12)	77
7	P1.5	58	29	P2.5(A13)	76
8	P1.6	59	30	P2.6(A14)	75
9	P1.7	60	31	P2.7(A15)	74
10	RST	61	32	/PSEN	73
11	P3.0(RXD)	62	33	ALE	72
12	NC		34	NC	
13	P3.1(TXD)	63	35	/EA	
14	P3.2(/INT0)	66	36	P0.7(AD7)	95
15	P3.3(/INT1)	67	37	P0.6(AD6)	94
16	P3.4(T0)	68	38	P0.5(AD5)	93
17	P3.5(T1)	69	39	P0.4(AD4)	88
18	P3.6(/WR)	70	40	P0.3(AD3)	87
19	P3.7(/RD)	71	41	P0.2(AD2)	86
20	XTAL2		42	P0.1(AD1)	85
21	XTAL1		43	P0.0(AD0)	84
22	GND		44	VCC/3.3V	

本實驗板的零件表

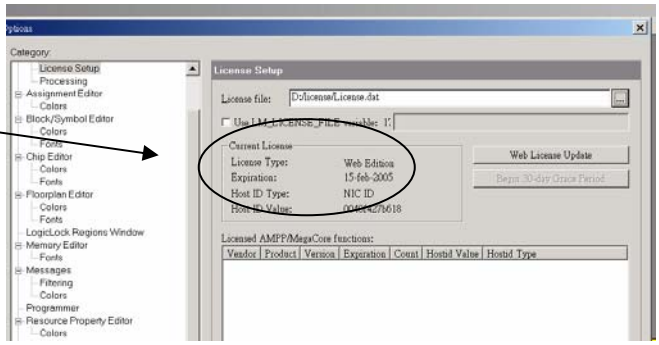
Component Reference No.	Feature
U1	Altera MaxII family device
U4	MPU 44 Pin PLCC Socket
J1,J2,J3,J4	Extension I/O Connectors
J5	DC JACK
J6	DC Input Header
J7	JTAG Header
J8	UART Connector
JP1	Clock Input Selector
D1,D2,D3,D4,D5,D6,D7,D8	LED
D9 , D10	UART Data Displayer
SW1,SW2,SW3,SW4	Push Button Switch
SW6	DIP Switch
DS1	4 Digital 7 Segment LED Displayer
Y1	Oscillator Socket
SW5	MPU Reset Push Button Switch
	Power LED

如何安裝 Quartus II 及 Down Load 的簡易操作方法

一. 安裝 Altera Quartus II 4.1 SP2 之後的版本，可打開隨附 CD，點選 Install Quartus II Web Edition

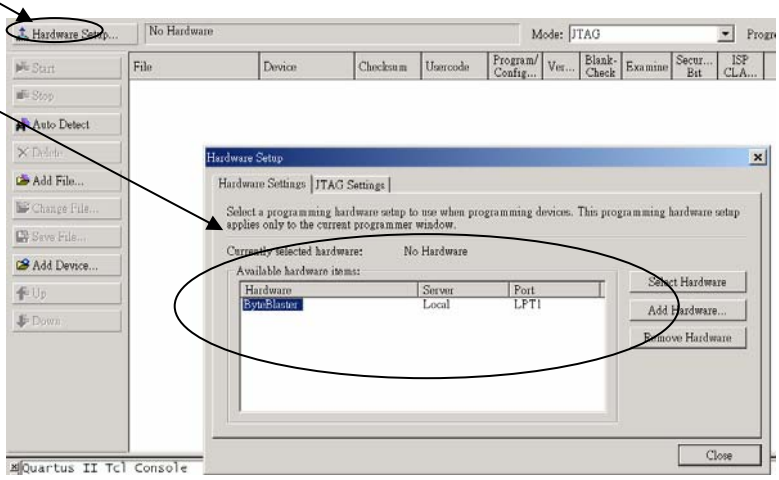
二. 申請 License

1. 打開 Quartus II 5.0(開始 => 程式集 => Altera => Quartus II 4.1)
2. 從主選單中的 Tools => License Seup => 抄下(NIC) ID
3. 再至 http://www.altera.com/support/licensing/lic-index.html，點選 Quartus II Web Edition Software License，即可向 Altera 申請一個 Free 的 Quartus II License
4. Altera 將經由 e-mail 回傳一個 License.dat 的檔案，打開 License Setup 的視窗，再經由 License file : 將 License.dat 的位置給它
5. 當出現右邊圖示後表示安裝完成



三. 當完成 Compiler 之後，再從 Tools => 打開 Programmer

1. 選擇 Hardware Setup
2. 選擇 ADD Hardware
   再按 OK
3. highlight ByteBlaster
   選擇 Select Hardware
   再按 close 即可



四. Mode => JTAG

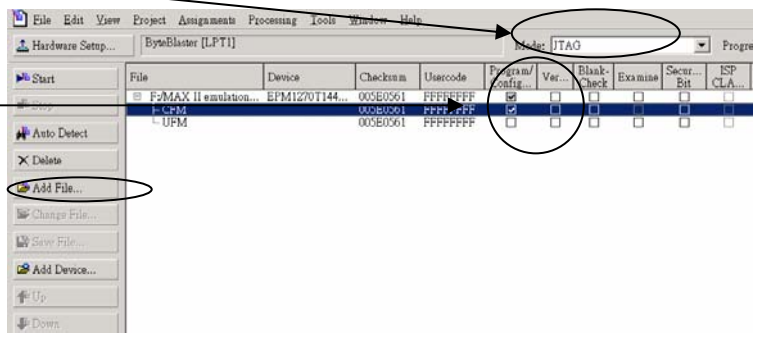
五. Add File，選擇 pof 檔

六. 若未使用 UFM，按照右邊
視窗打勾

七. 再將 Download
cable 插進 J7

八. 專用變壓器插進 J6

九. 再按 Start 即可燒錄

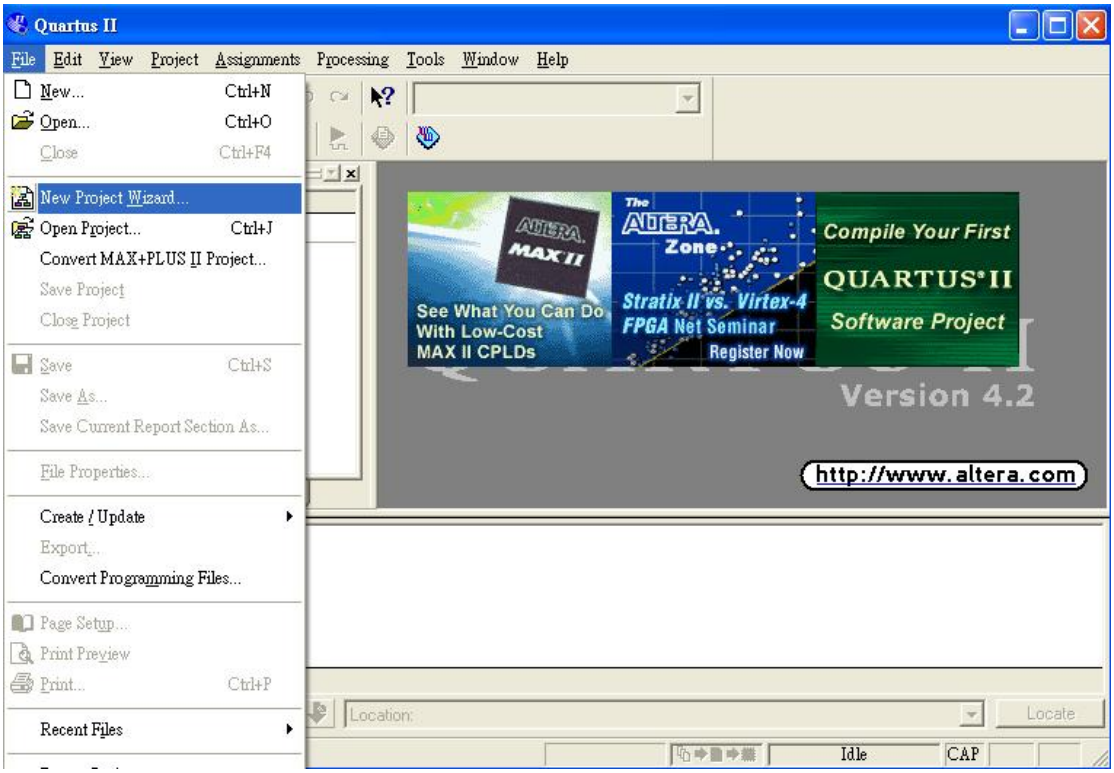


第二章　Quartus II 快速入門

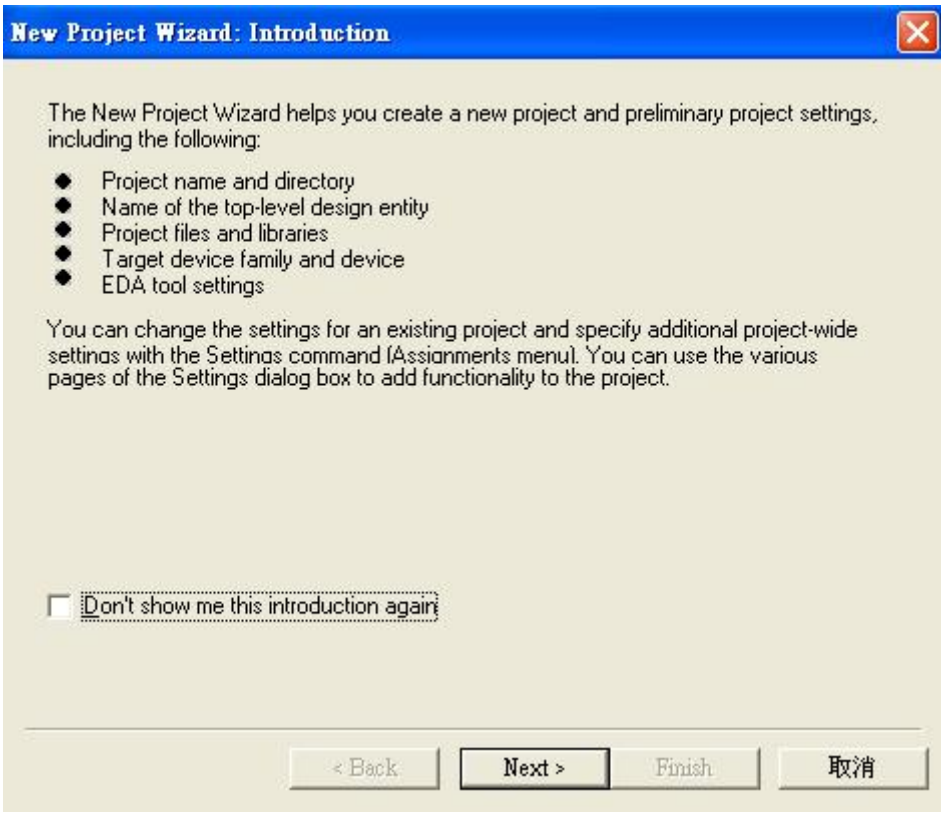
在本章節中將以一個 VHDL 設計的跑馬燈範例示範如何以 Quartus II 進行新增專案、程式設計、編譯與下載等流程。在開始進行下面的步驟之前必須將光碟中的 ledtest 資料夾複製到硬碟中，以下範例是將 ledtest 複製到 C:\altera\qdesigns42 路徑下。

Part 1. 新增專案

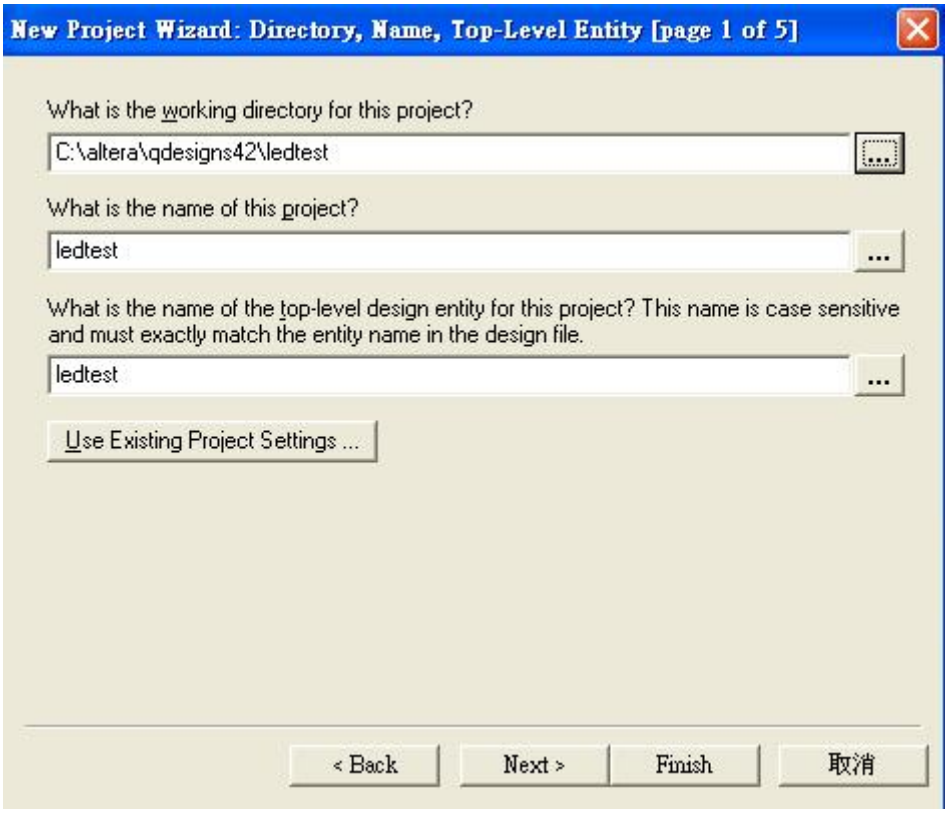
步驟1.　開啟專案精靈，【File】→【New Project Wizard】



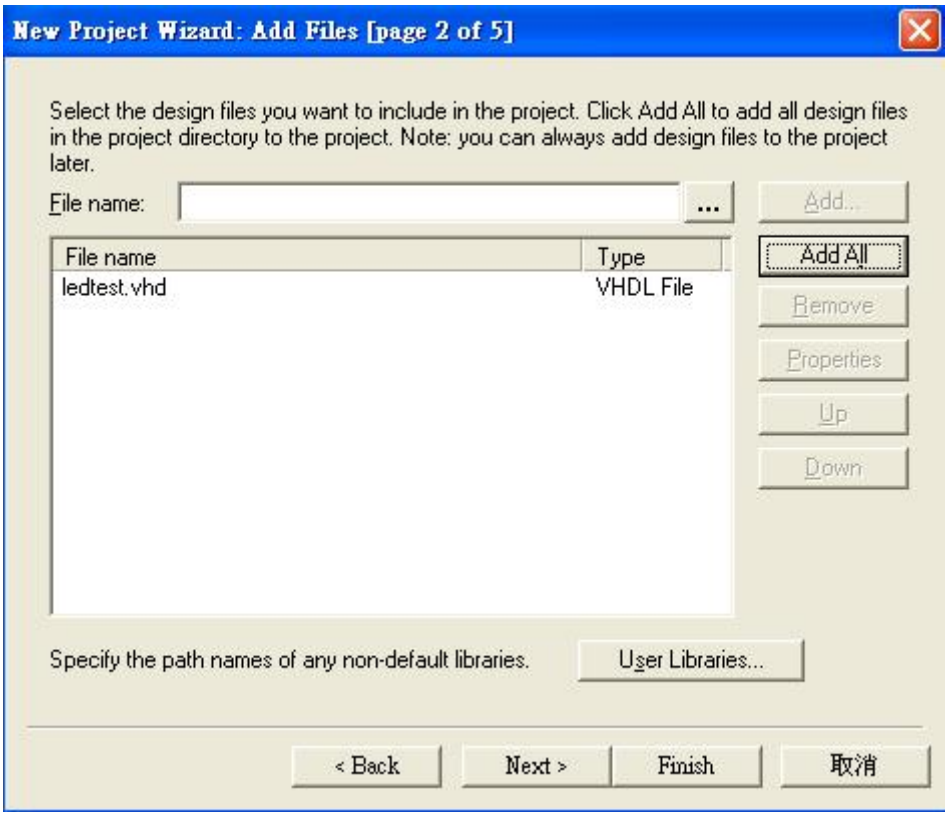
步驟2.　按【Next】



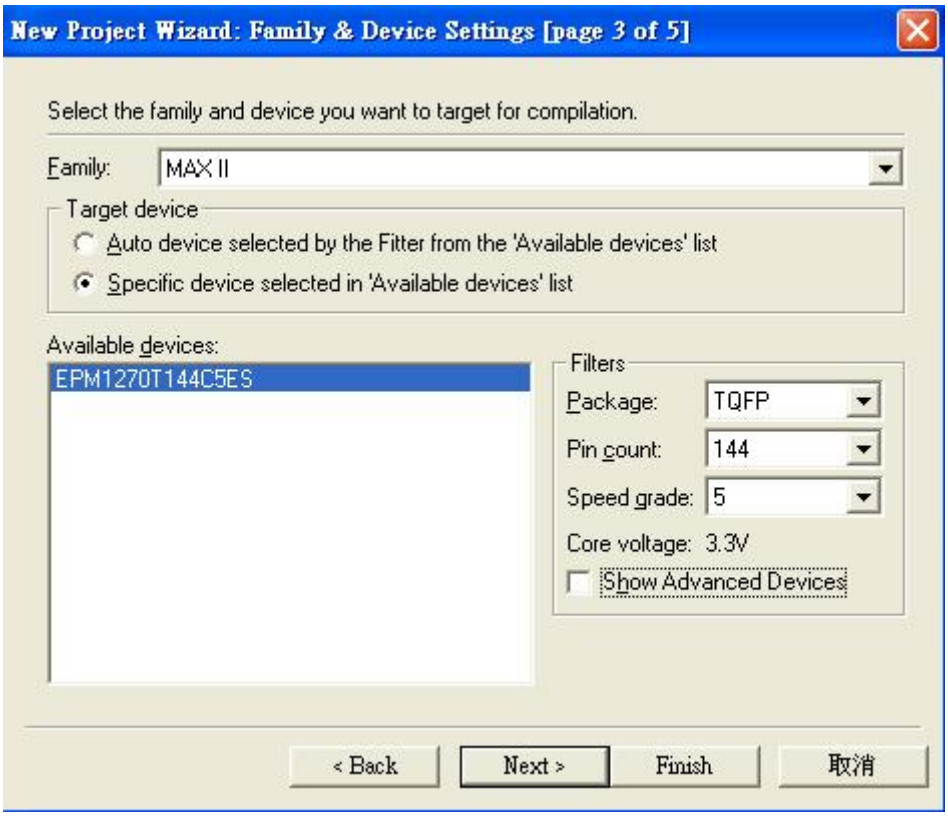
步驟3.　輸入專案工作目錄與名稱，按【Next】



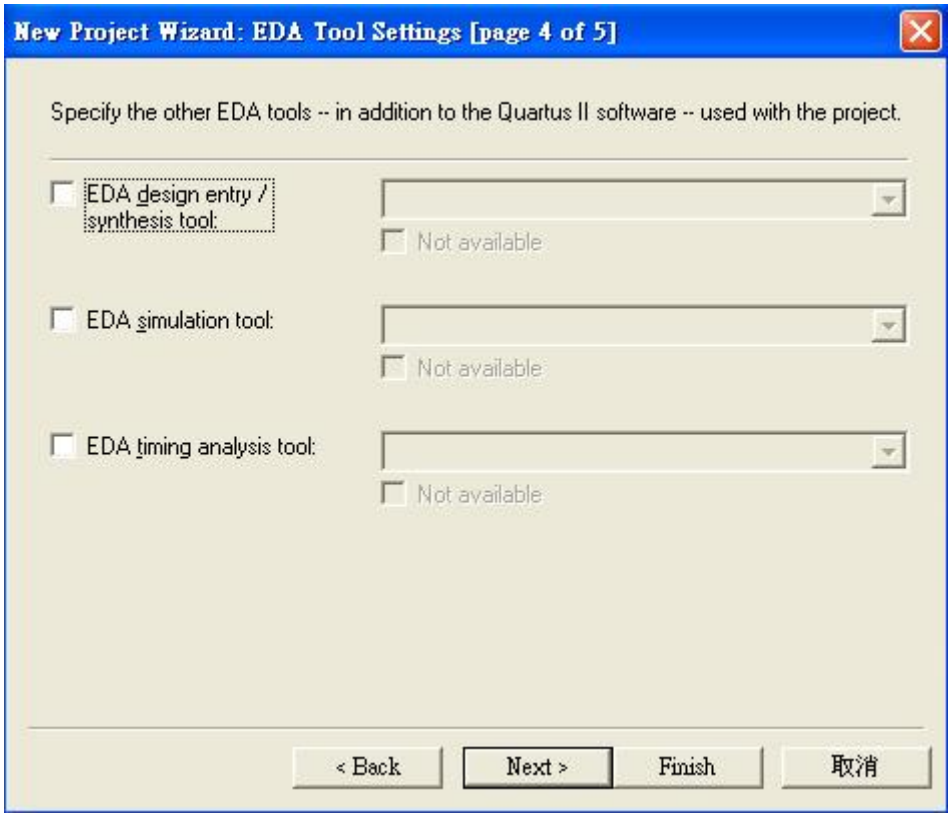
步驟4.　按【Add All】將 ledtest.vhd 加入此專案中，按【Next】


New Project Wizard: Add Files [page 2 of 5]
Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project. Note: you can always add design files to the project later.
File name:
...
Add...
Add All
Remove
Properties
Up
Down
File name
Type
ledtest.vhd
VHDL File
Specify the path names of any non-default libraries.
User Libraries...
< Back
Next >
Finish
取消

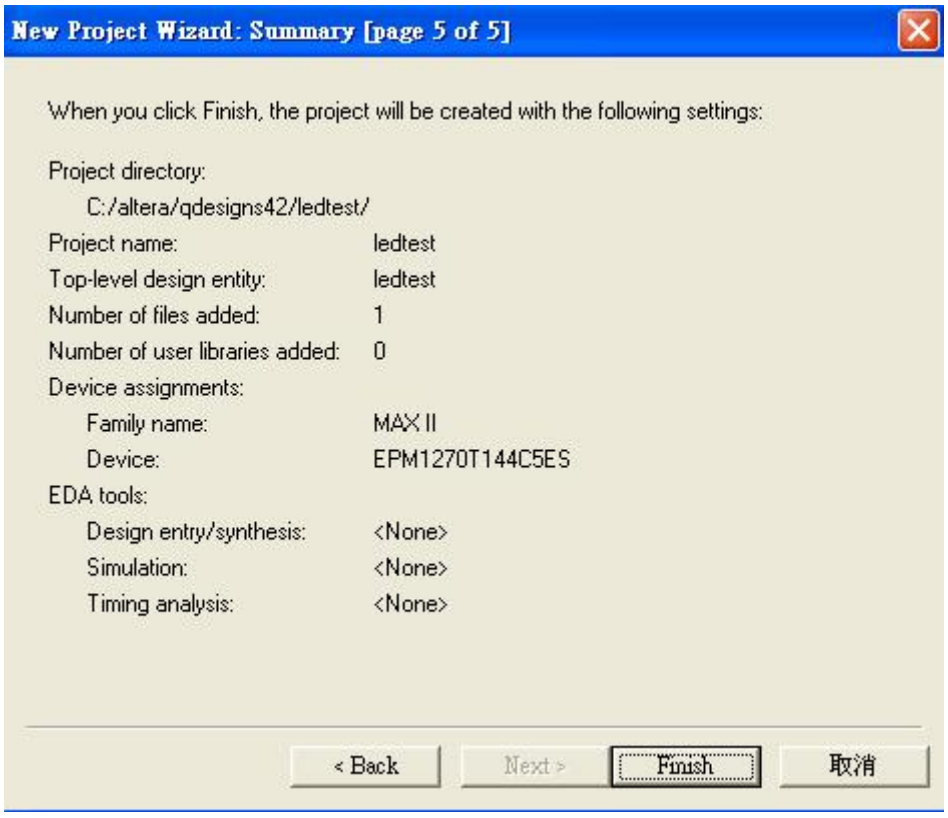
步驟5.　選擇晶片型號，選擇 MAX II 系列的 EPM1270T144C5ES



步驟6.　按【Next】

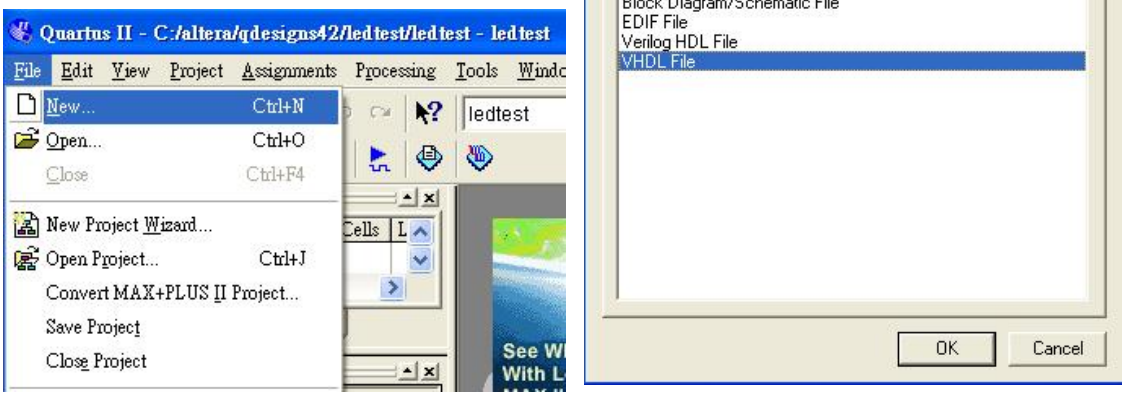


步驟7.　按【Finish】完成新增專案

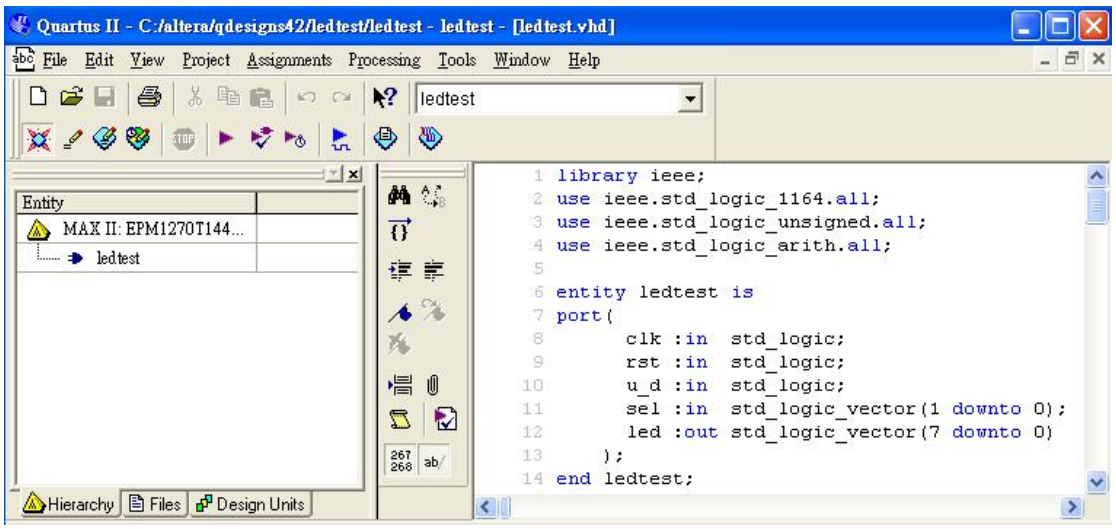


Part 2. 建立程式檔

Quartus II 可以用來輸入電路的方式有 AHDL、Block Diagram/Schematic、EDIF、Verilog HDL、VHDL File 等等，在此章節我們介紹以 VHDL 的方式來完成跑馬燈程式範例。要建立一個新的 VHDL 程式可點選【File】→【New】開啟設計方式選項，在【Device Design Files】下點選【VHDL File】即可開啟一個新的 VHDL 設計檔案。因為本範例已經提供一個以設計好的程式碼，所以只需用開啟舊檔的方式開啟 ledtest.vhd 檔。



步驟8.　開啟檔案 ledtest.vhd



跑馬燈程式碼與說明

【跑馬燈程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity ledtest is
7    port(
8        clk :in  std_logic;
9        rst :in  std_logic;
10       u_d :in  std_logic;
11       sel :in  std_logic_vector(1 downto 0);
12       led :out std_logic_vector(7 downto 0)
13     );
14 end ledtest;
15
16 architecture a of ledtest  is
17   signal cnt   : std_logic_vector(2 downto 0);
18   signal clkcnt: std_logic_vector(19 downto 0);
19   signal clk2  : std_logic;
20   signal led_t1,led_t2,led_t3,led_t4:std_logic_vector(7 downto 0);
21 begin
22
23   process(rst,clk)
24   begin
25     if rst='0' then
26       clkcnt<=(others=>'0');
27       clk2<='0';
28     elsif clk'event and clk='1' then
29       if clkcnt="11110100001001000000" then
30         clk2<=not clk2;
31         clkcnt<=(others=>'0');
32       else
33         clkcnt<=clkcnt+1;
34       end if;
35     end if;
36   end process;
37
38   process(rst,clk2)
39   begin
40     if rst='0' then
41       cnt<="000";
42     elsif clk2'event and clk2='1' then
43       if u_d='1' then
44         cnt<=cnt+1;
45       else
46         cnt<=cnt-1;
47       end if;
48     end if;
49   end process;
50
51   led_t1<="11111110" when cnt="000" else
52          "11111101" when cnt="001" else
53          "11111011" when cnt="010" else
54          "11110111" when cnt="011" else
```

```
55          "11101111" when cnt="100" else
56          "11011111" when cnt="101" else
57          "10111111" when cnt="110" else
58          "01111111";
59
60    led_t2<="01111110" when cnt="000" else
61          "10111101" when cnt="001" else
62          "11011011" when cnt="010" else
63          "11100111" when cnt="011" else
64          "01111110" when cnt="100" else
65          "10111101" when cnt="101" else
66          "11011011" when cnt="110" else
67          "11100111";
68
69    led_t3<="00000000" when cnt="000" else
70          "11111111" when cnt="001" else
71          "00000000" when cnt="010" else
72          "11111111" when cnt="011" else
73          "00000000" when cnt="100" else
74          "11111111" when cnt="101" else
75          "00000000" when cnt="110" else
76          "11111111";
77
78    led_t4<="11101110" when cnt="000" else
79          "11011101" when cnt="001" else
80          "10111011" when cnt="010" else
81          "01110111" when cnt="011" else
82          "10111011" when cnt="100" else
83          "11011101" when cnt="101" else
84          "11101110" when cnt="110" else
85          "11111111";
86
87    led <= led_t2 when sel="10" else
88          led_t3 when sel="01" else
89          led_t4 when sel="00" else
90          led_t1;
91 end a;
```

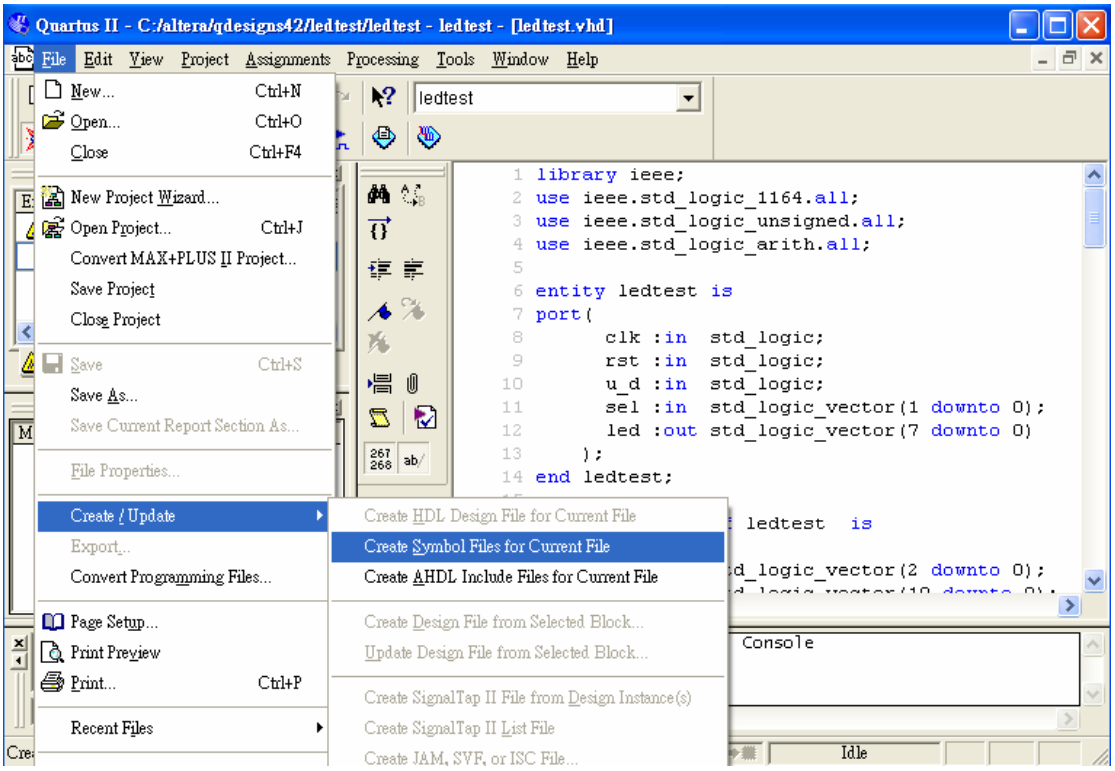
◎跑馬燈程式碼說明

行　號	說　明
1~4	元件庫宣告
6~14	輸入與輸出埠宣告
16~21	內部訊號宣告
23~36	除頻電路，將 16MHz 的系統頻率降為 8Hz 的跑馬燈頻率 clk2

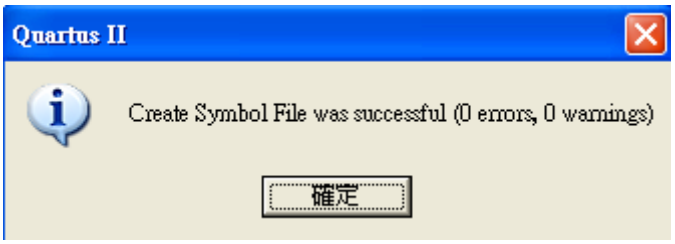
38~49	3 位元上下數計數器
51 ~ 85	四種解碼電路，產生四種不同變化的跑馬燈訊號
87~90	多工電路，以 sel 來選擇以哪種變化訊號當做輸出顯示在 LED 上

我們可以將此 VHDL 檔建成一個符號(symbol)，以供畫電路圖的方式設計電路，建立符號的流程如下：

Step1. 開啟要建立符號的 VHDL 檔，點選【File】→【Creat/Update】→【Create SymbolFiles for Current File】

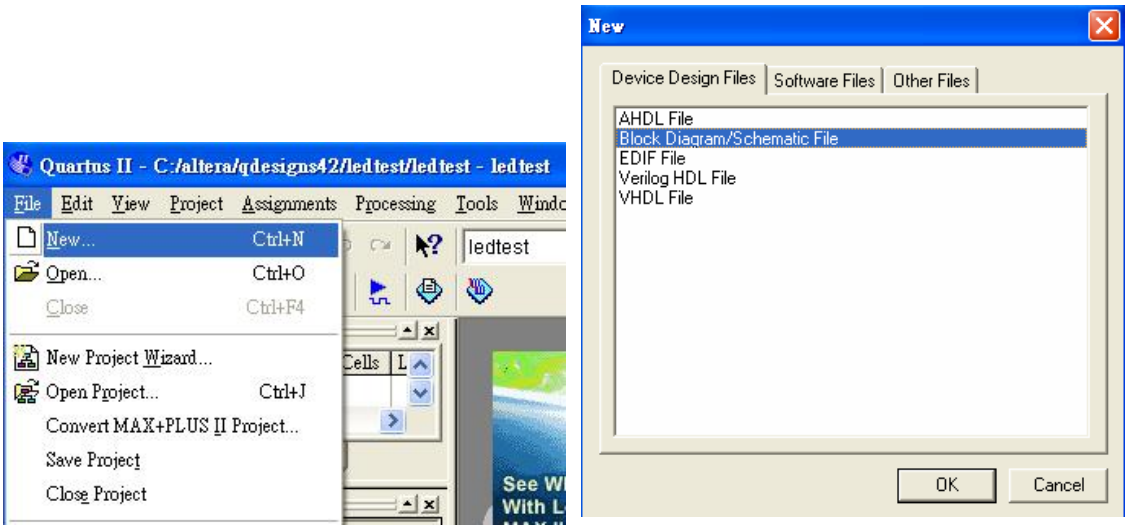


Step2. 完成符號建立

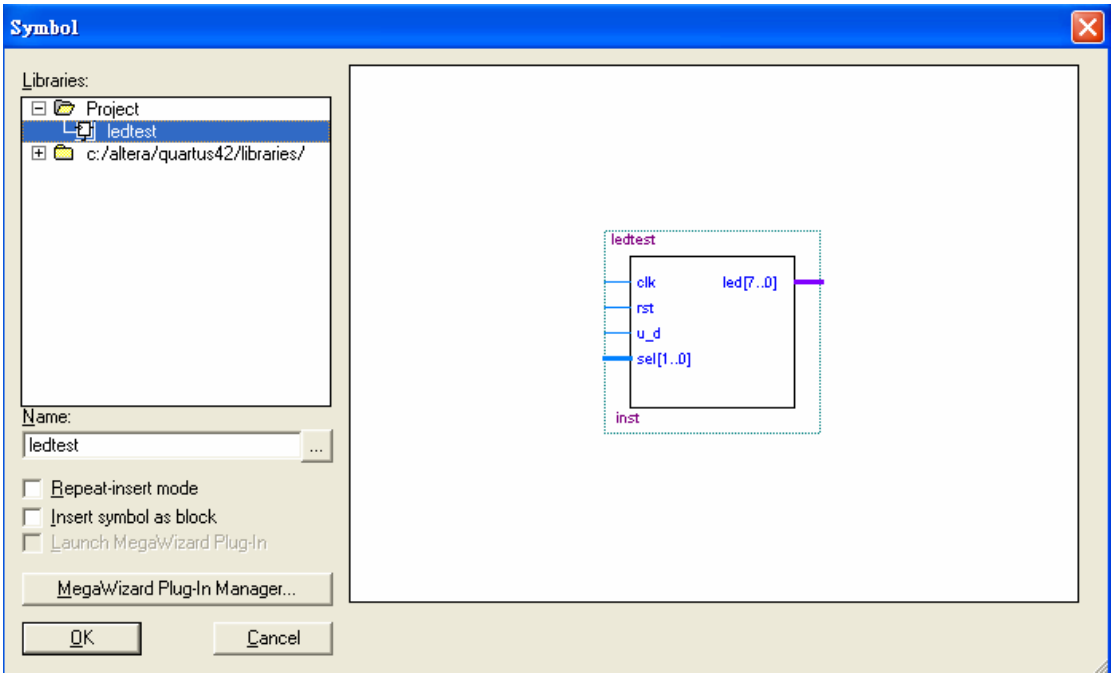


Step3. 要建立一個新的電路圖檔(Block Diagram File)可點選【File】→【New】

開啟設計方式選項，在【Device Design Files】下點選【Block Diagram/Schematic File】即可開啟一個新的電路圖設計檔案



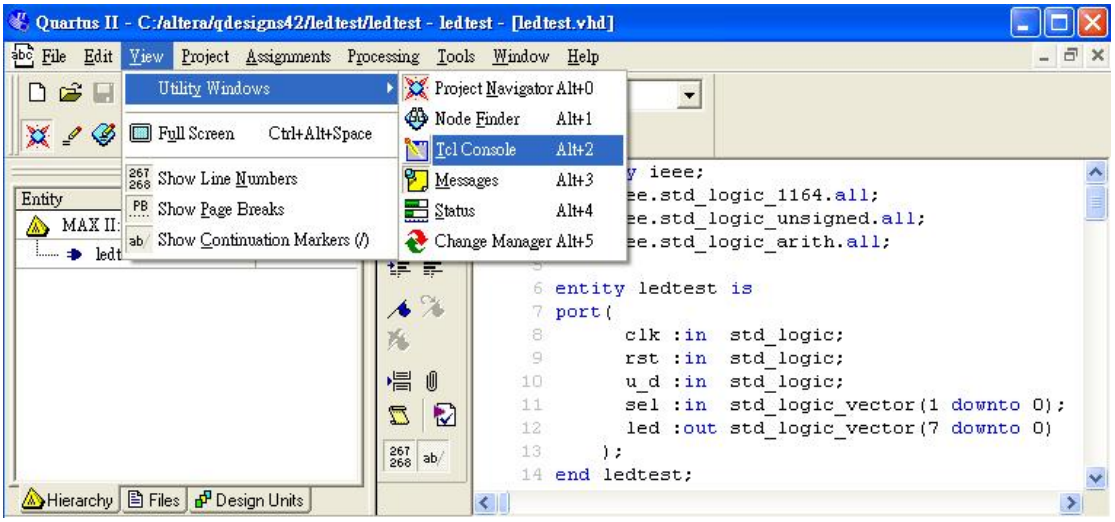
Step4. 開啟新的電路圖檔後，在空白處按滑鼠左鍵兩下，即可開啟符號選擇視窗，在【Project】下即可看到已經建立好的 ledtest 符號



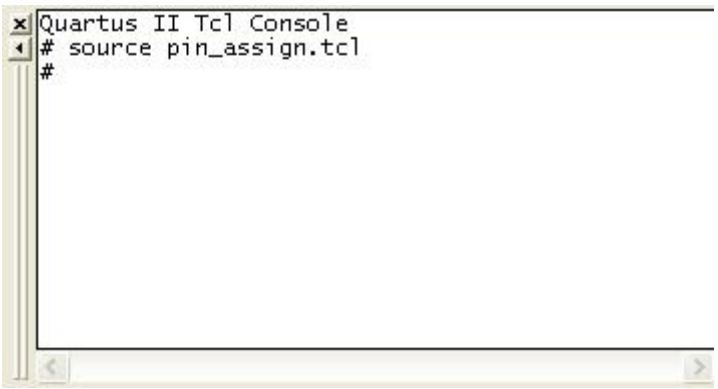
Part 3. 指定晶片腳位

Quartus II 提供幾種方式來指定腳位，可以在【Assignments】→【Pins】選單中逐一指定，也可以利用 Tcl script 來設定，在本範例中將介紹如何使用 Tcl script 來指定腳位。

步驟9.　開啟 Tcl Console 畫面，【View】→【Utility Windows】→【Tcl Console】



步驟10.　在 Tcl Console 視窗下輸入 source pin_assign.tcl 後按 enter，完成腳位指定



Tcl 腳位設定程式碼與說明

【pin_assign.tcl 程式碼】

```
1  ##############################################################################
2  # pin_assign.tcl
3  #
4  # This script allows you to make pin assignments to the Ledtest design
5  #
6  # Written by: Hans.Lin
7  # Rev 1.0
8  # 2005/04/10
9  #
10 # You can run this script from Quartus by observing the following steps:
11 # 1. Place this TCL script in your project directory
12 # 2. Open your project
13 # 3. Go to the View Menu and Auxilary Windows -> TCL console
```

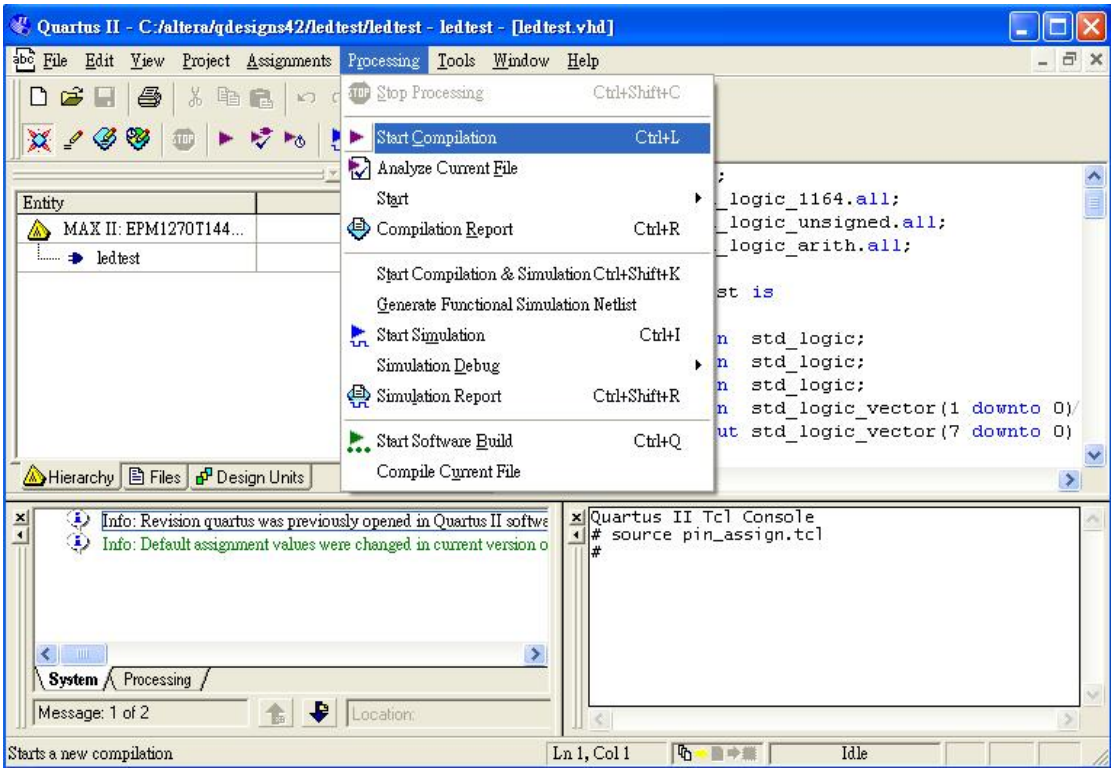
```
14 # 4. In the TCL console type:
15 #                    source pin_assign.tcl
16 # 5. The script will assign pins.
17 ##########################################################################
18 set_location_assignment PIN_18 -to clk
19 set_location_assignment PIN_23 -to rst
20 set_location_assignment PIN_24 -to u_d
21 set_location_assignment PIN_27 -to sel\[0\]
22 set_location_assignment PIN_28 -to sel\[1\]
23 set_location_assignment PIN_1 -to led\[0\]
24 set_location_assignment PIN_2 -to led\[1\]
25 set_location_assignment PIN_3 -to led\[2\]
26 set_location_assignment PIN_4 -to led\[3\]
27 set_location_assignment PIN_5 -to led\[4\]
28 set_location_assignment PIN_6 -to led\[5\]
29 set_location_assignment PIN_7 -to led\[6\]
30 set_location_assignment PIN_8 -to led\[7\]
```

◎pin_assign.tcl 程式碼說明

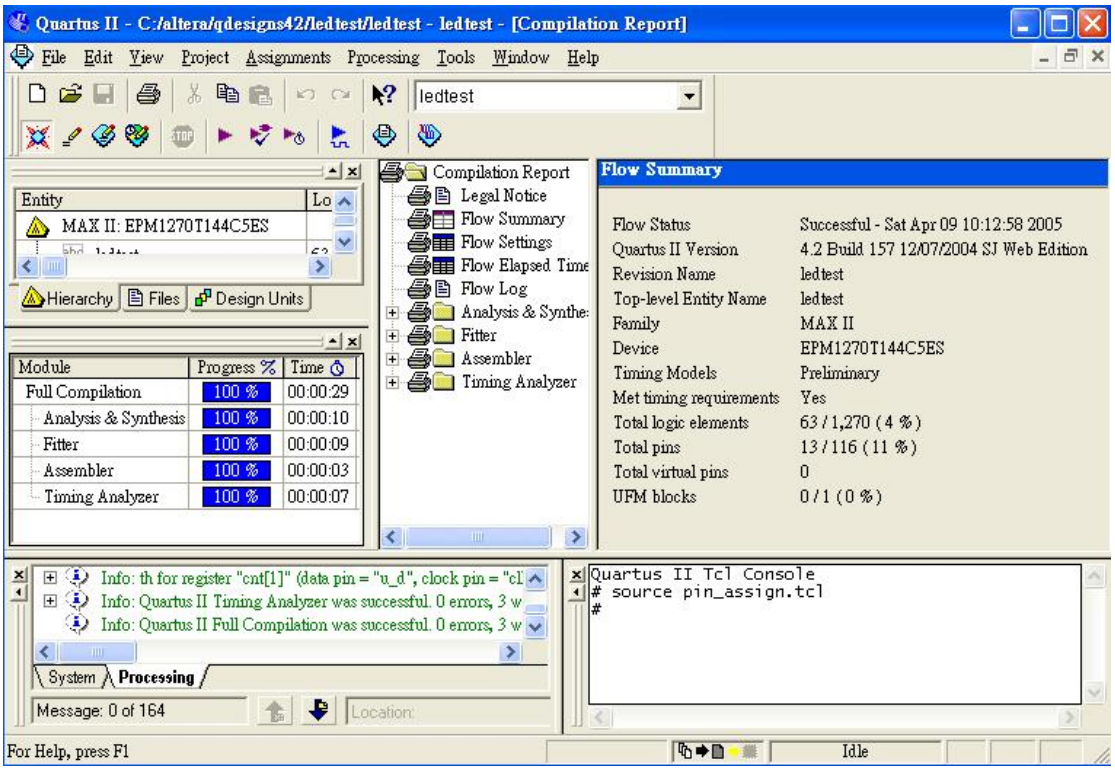
行　號	說　明
1~17	註解
18~30	腳位指定

Part 4. 程式編譯

步驟11. 進行編譯，【Processing】→【Start Compilation】

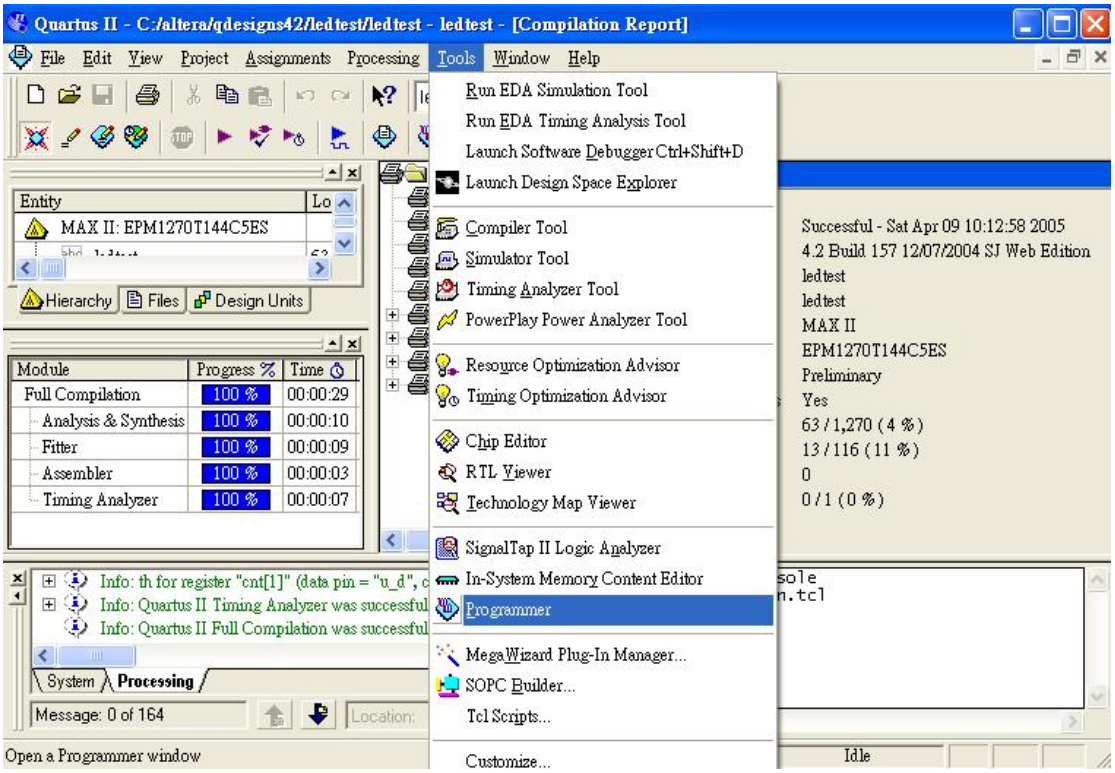


步驟12. 完成編譯

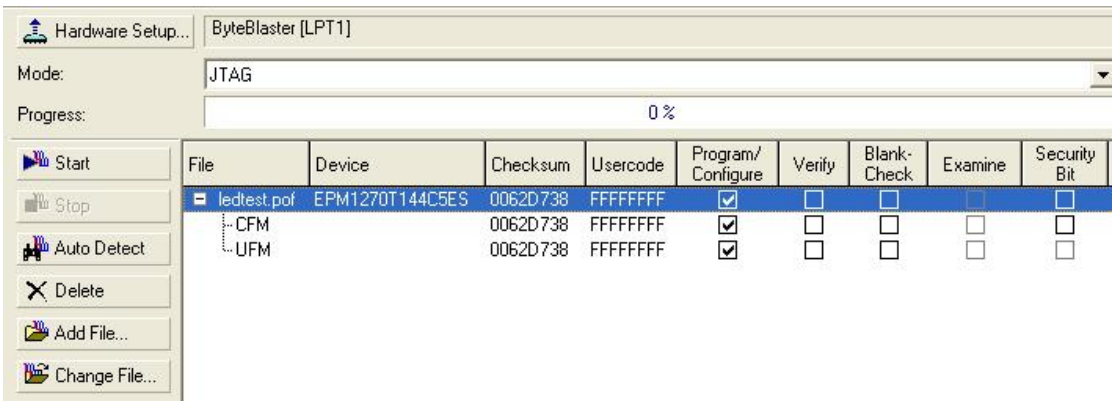

Quartus II - C:/altera/qdesigns42/ledtest/ledtest - ledtest - [Compilation Report]
File Edit View Project Assignments Processing Tools Window Help
ledtest
Entity
MAX II: EPM1270T144C5ES
Hierarchy Files Design Units
Module Progress Time
Full Compilation 100 % 00:00:29
Analysis & Synthesis 100 % 00:00:10
Fitter 100 % 00:00:09
Assembler 100 % 00:00:03
Timing Analyzer 100 % 00:00:07
Compilation Report
Legal Notice
Flow Summary
Flow Settings
Flow Elapsed Time
Flow Log
Analysis & Synthesis
Fitter
Assembler
Timing Analyzer
Flow Summary
Flow Status Successful - Sat Apr 09 10:12:58 2005
Quartus II Version 4.2 Build 157 12/07/2004 SJ Web Edition
Revision Name ledtest
Top-level Entity Name ledtest
Family MAX II
Device EPM1270T144C5ES
Timing Models Preliminary
Met timing requirements Yes
Total logic elements 63 / 1,270 (4 %)
Total pins 13 / 116 (11 %)
Total virtual pins 0
UFM blocks 0 / 1 (0 %)
Info: th for register "cnt[1]" (data pin = "u_d", clock pin = "cl
Info: Quartus II Timing Analyzer was successful. 0 errors, 3 w
Info: Quartus II Full Compilation was successful. 0 errors, 3 w
System Processing
Message: 0 of 164
Location:
Quartus II Tcl Console
source pin_assign.tcl
#
For Help, press F1
Idle

Part 5. 下載程式

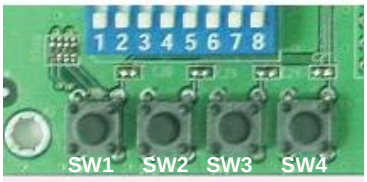
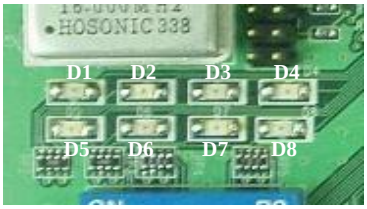
步驟13. 開啟 Programmer，【Tools】→【Programmer】



步驟14. 確認已經安裝好 Download Cable，若沒有的話請參閱『Windows XP/2000 下安裝 Download Cable』，Hardware Setup 選擇為 ByteBlaster[LPT1]，若不是的話請參閱『Hardware Setup』一節。下載檔案選擇 ledtest.pof，勾選【Program/Configure】，此時 MAX II Kit 必須接上 Download Cable 下載線與電源線，最後按下【Start】完成下載動作。

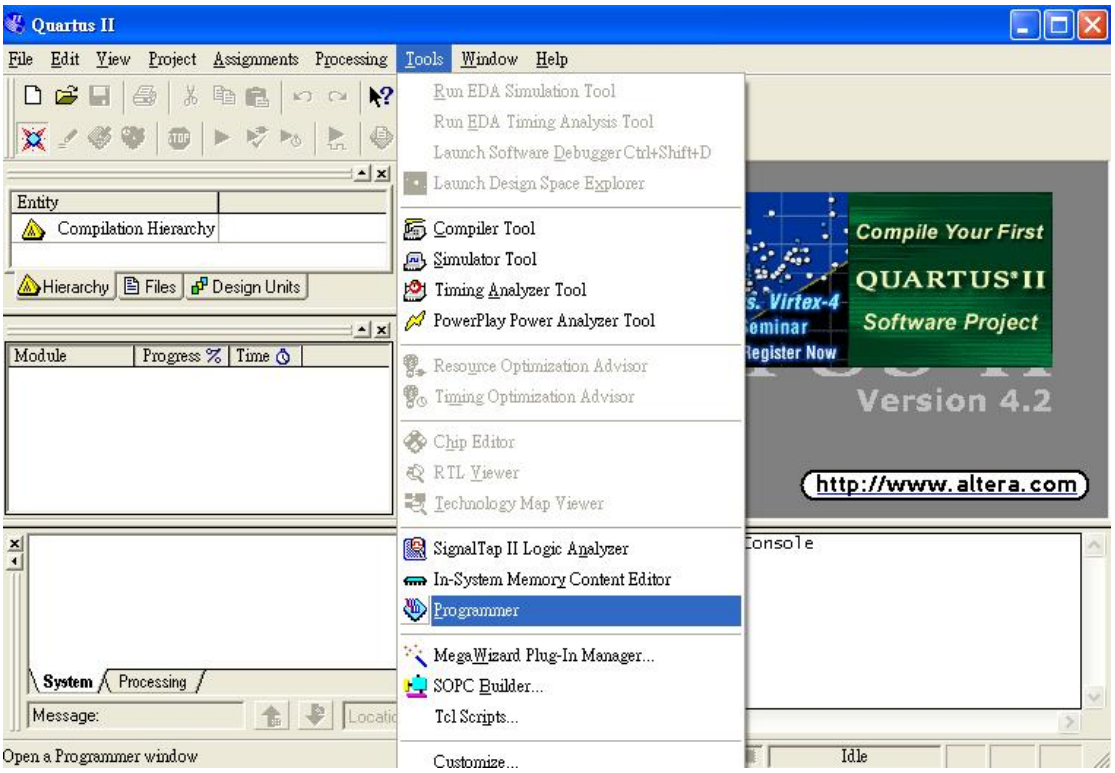


Part 6. 硬體設定與操作說明

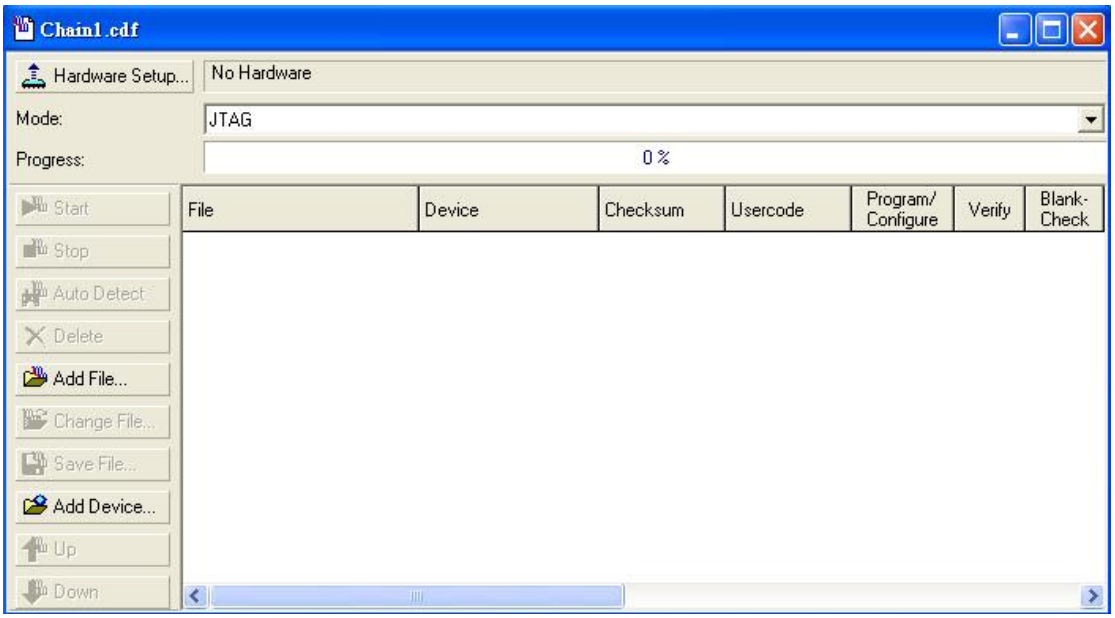
<table>
<tr><th>硬體介面</th><th>說明</th></tr>
<tr><td>
</td><td>➢ Jumper 插上 JP1 的第一組提供系統頻率至 MAX II Pin 18</td></tr>
<tr><td> 
</td><td>➢ SW1：按下時為重置功能
➢ SW2：上下計數控制鍵，按下時 LED 亮的順序會相反
➢ SW3,SW4：跑馬燈種類選擇，共有四種選擇
<table>
<tr><td>SW3</td><td>SW4</td><td>跑馬燈</td></tr>
<tr><td>OFF</td><td>OFF</td><td>依序亮一個 LED</td></tr>
<tr><td>OFF</td><td>ON</td><td>全亮全滅不斷閃爍</td></tr>
<tr><td>ON</td><td>OFF</td><td>以繞圈順序亮兩個 LED</td></tr>
<tr><td>ON</td><td>ON</td><td>依序上下兩個 LED 同時亮</td></tr>
</table></td></tr>
<tr><td> 
</td><td>➢ 跑馬燈輸出顯示至此 8 個 LED</td></tr>
</table>

Part 7. Hardware Setup

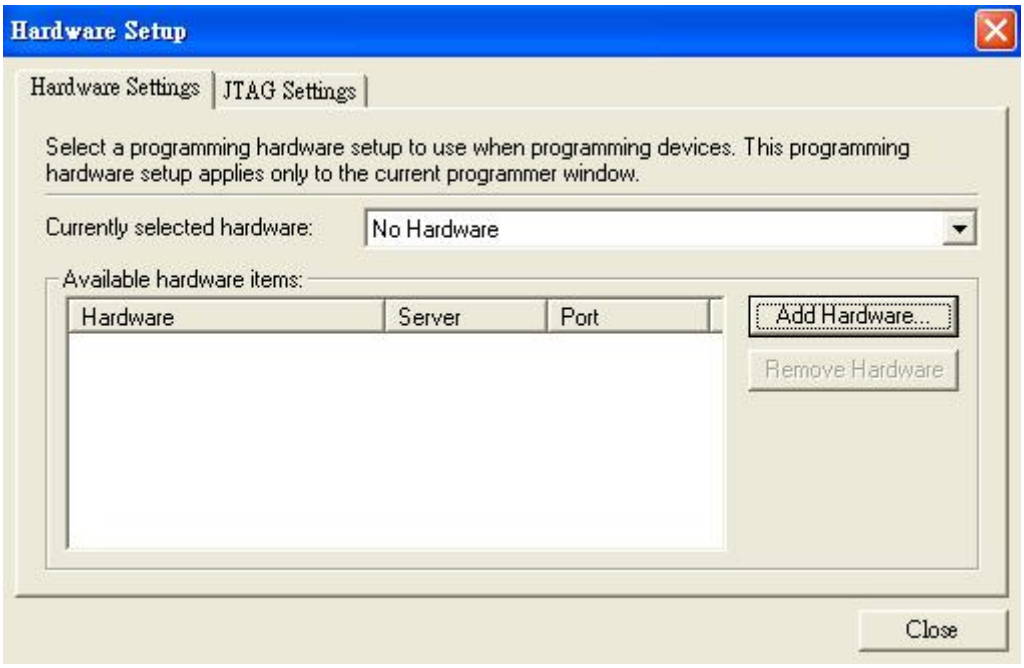
步驟1.　【Tools】→【Programmer】



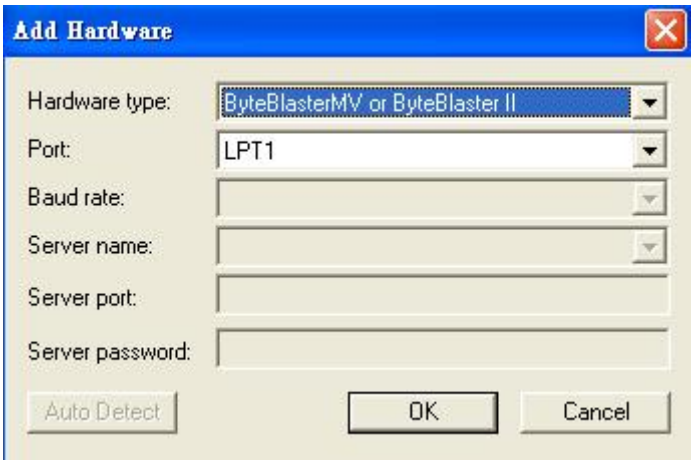
步驟2.　按【Hardware Setup】



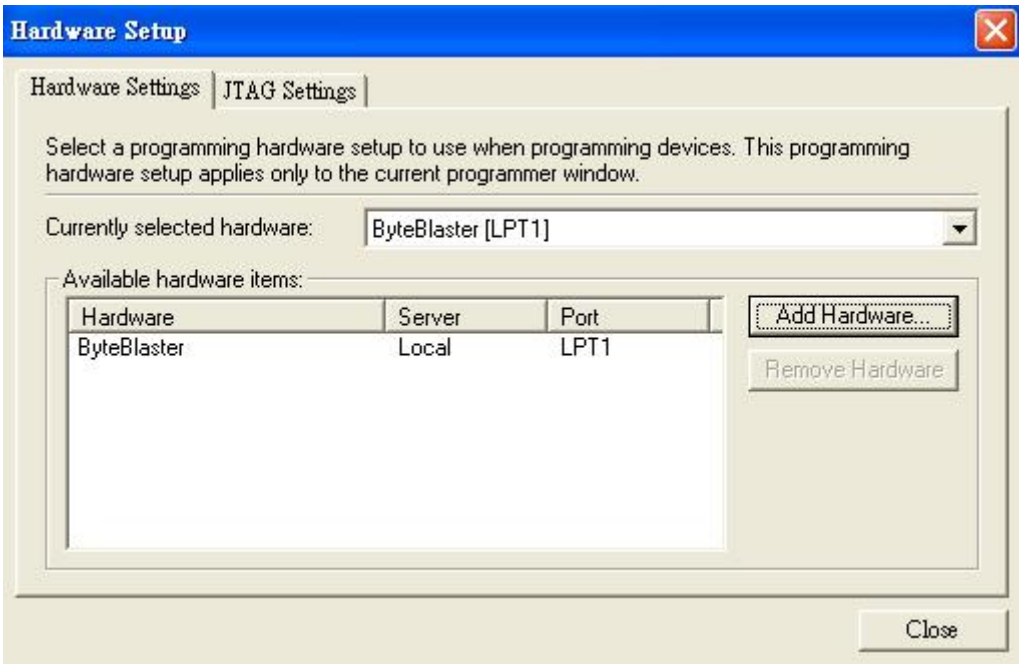
步驟3.　按【Add Hardware】



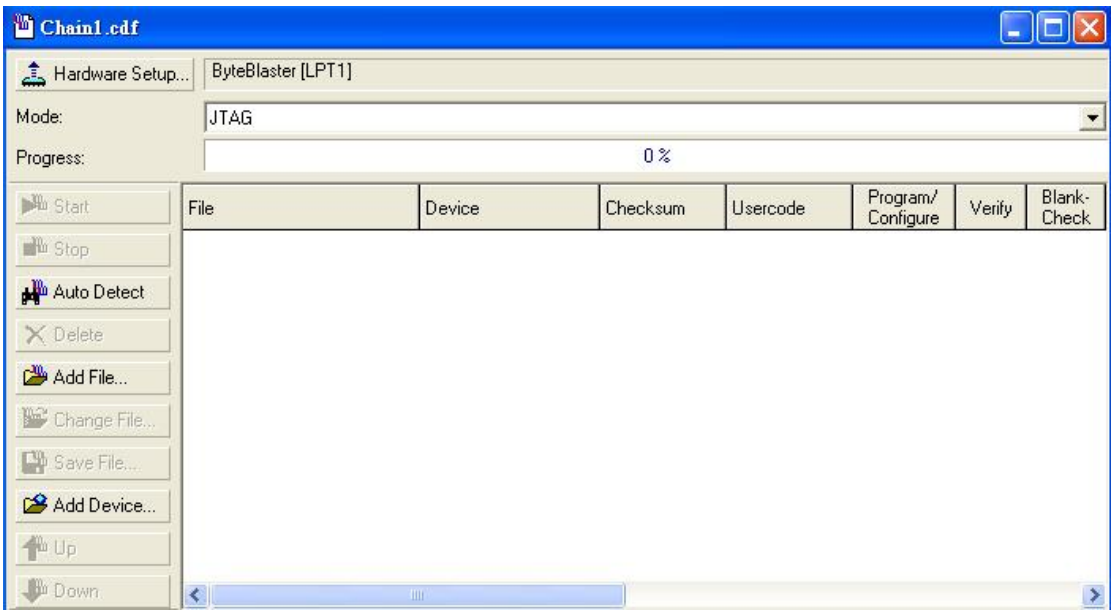
步驟4.　Hardware type 選擇 ByteBlasterMV or ByteBlaster II，Port 選擇 LPT1，按下【OK】



步驟5.　按【Close】



步驟6.　完成 Hardware Setup



第三章　測試程式下載說明

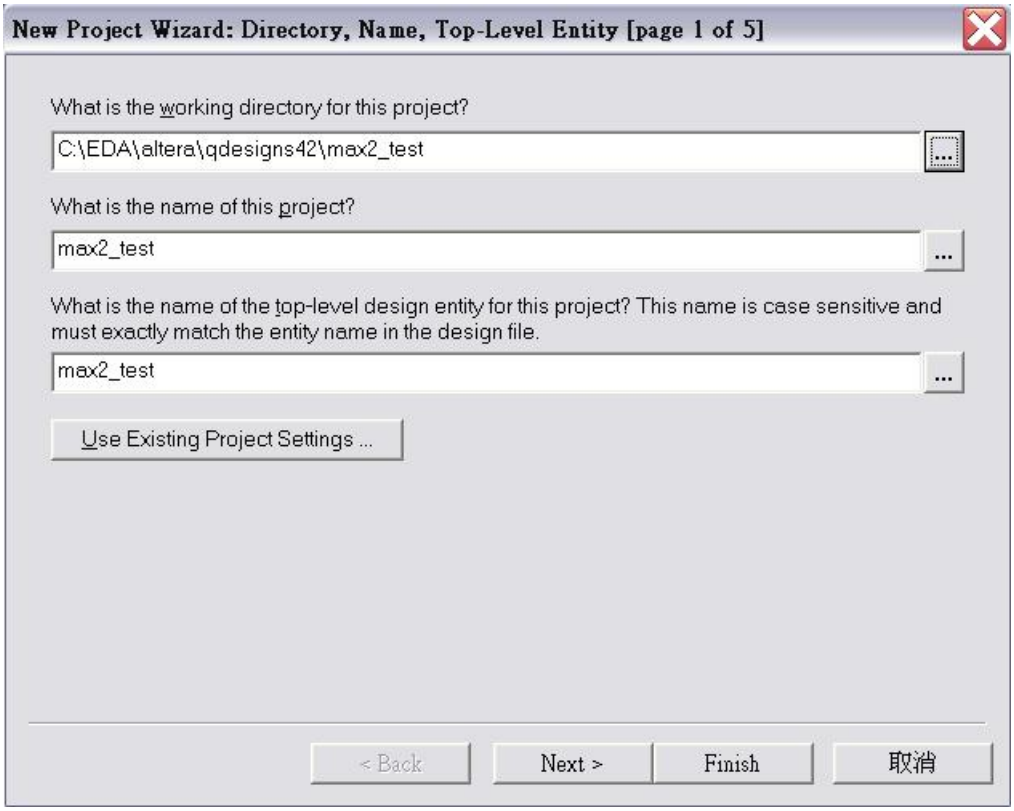
此章節說明如何測試 MAX II 實驗板上的介面是否正常，將由測試程式碼與腳位規劃檔以 QuartusII 編譯並下載燒錄檔到 MAX II 中。

步驟1. 硬體設定：將 MAX II 實驗板接上電源與 Download cable，將 Download cable 程式下載線接至電腦主機的 Printer Port 1，將 JP1 接至 1, 2 接點

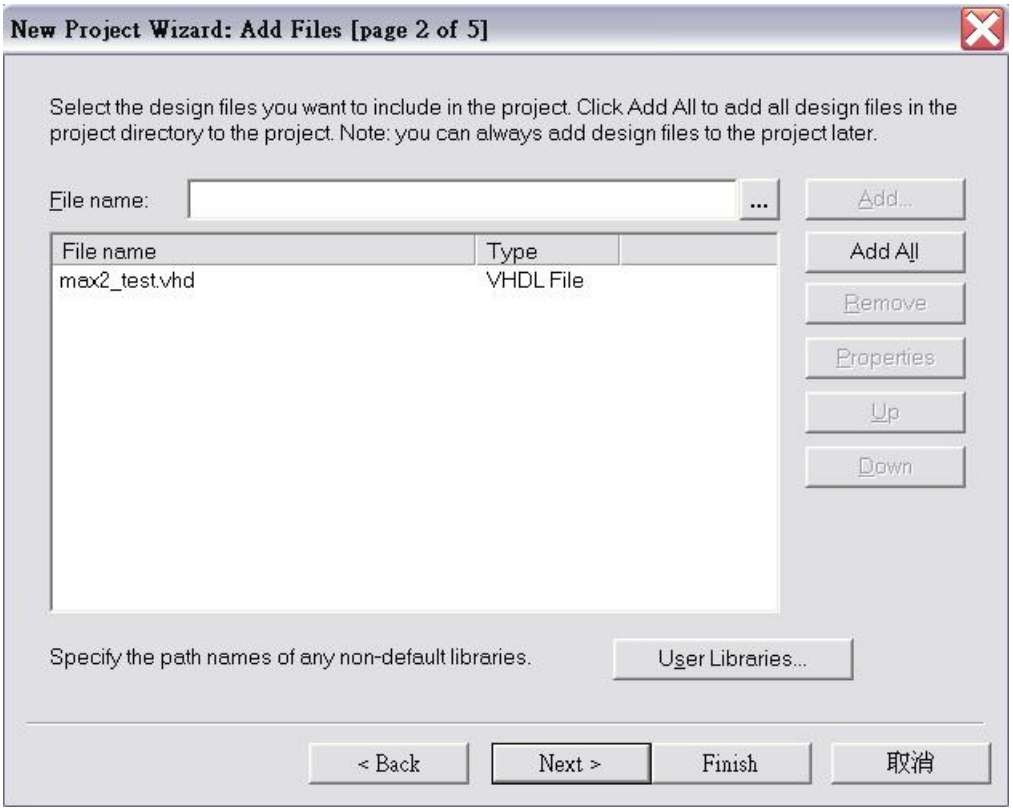
步驟2. 將光碟中的 max2_test 資料夾複製於您的硬碟中，並開啟 Quartus II

步驟3. 點選【File】→【New Project Wizard】，開啟建立專案精靈視窗

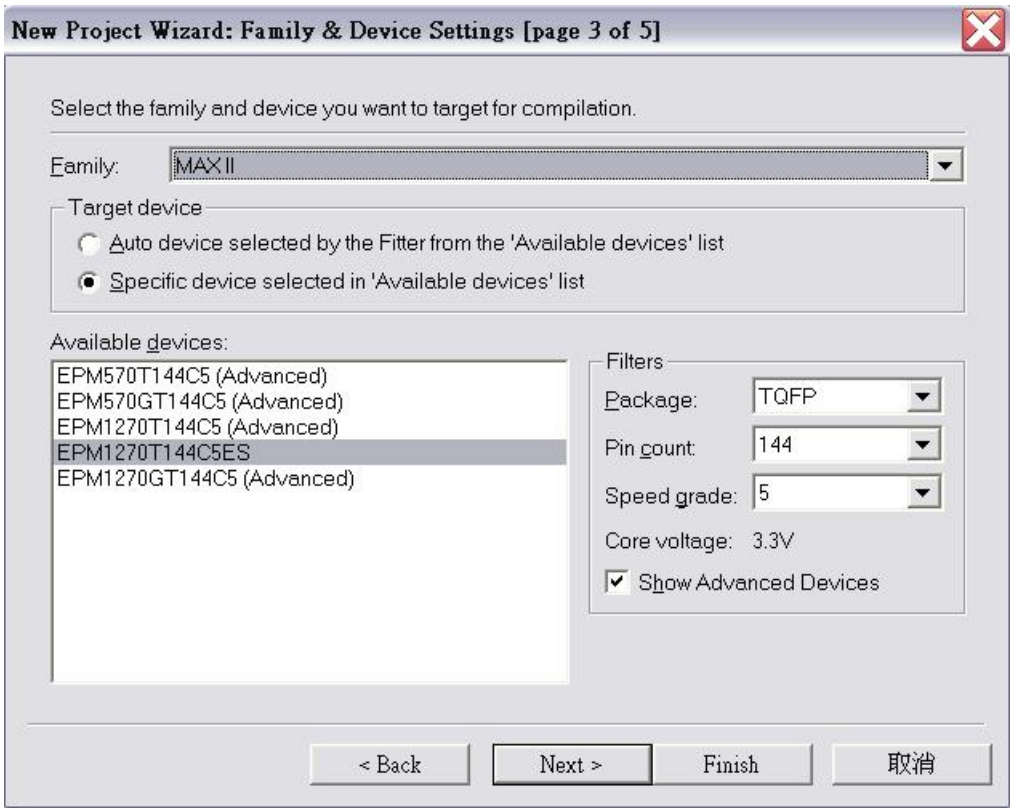
步驟4. 將專案的工作路徑指定到 max2_test 資料夾，也就是步驟 2 時您所存放 max2_test 資料夾於您的硬碟中的位址路徑，按【Next】


New Project Wizard: Directory, Name, Top-Level Entity [page 1 of 5]
What is the working directory for this project?
C:\EDA\altera\qdesigns42\max2_test
What is the name of this project?
max2_test
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.
max2_test
Use Existing Project Settings ...
< Back
Next >
Finish
取消

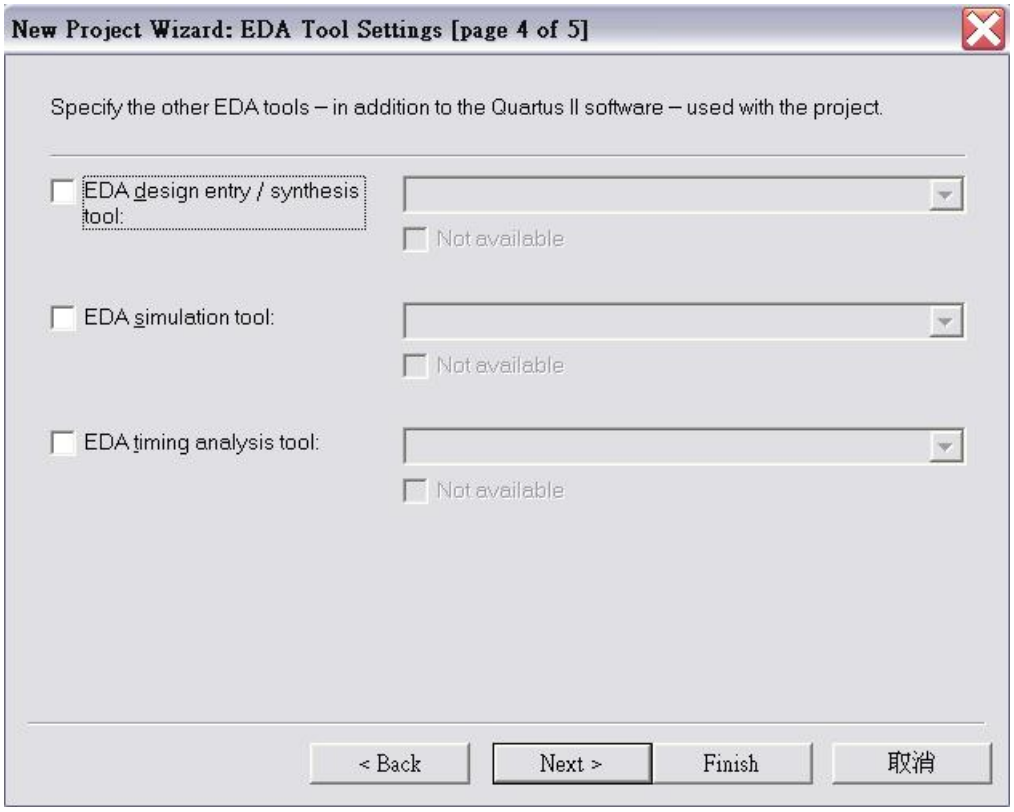
步驟5. 按【Add All】將 max2_test.vhd 加入到專案中，按【Next】


New Project Wizard: Add Files [page 2 of 5]
Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project. Note: you can always add design files to the project later.
File name:
Add...
Add All
Remove
Properties
Up
Down
File name
Type
max2_test.vhd
VHDL File
Specify the path names of any non-default libraries.
User Libraries...
< Back
Next >
Finish
取消

步驟6. 選取晶片型號，Family：選取 MAX II；指定晶片為 TQFP 包裝，144 Pin count，5 Speed grade，晶片型號為 EPM1270T144C5ES

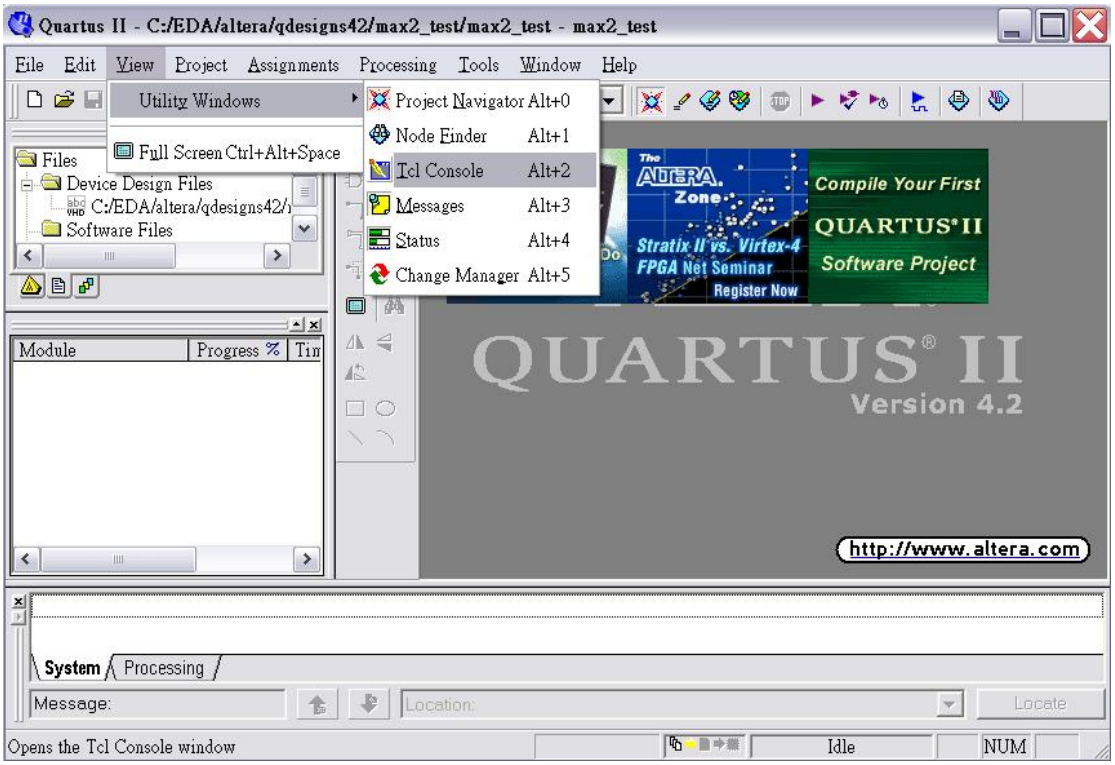


步驟7. 按【Next】


New Project Wizard: EDA Tool Settings [page 4 of 5]
Specify the other EDA tools – in addition to the Quartus II software – used with the project.
EDA design entry / synthesis tool:
Not available
EDA simulation tool:
Not available
EDA timing analysis tool:
Not available
< Back
Next >
Finish
取消

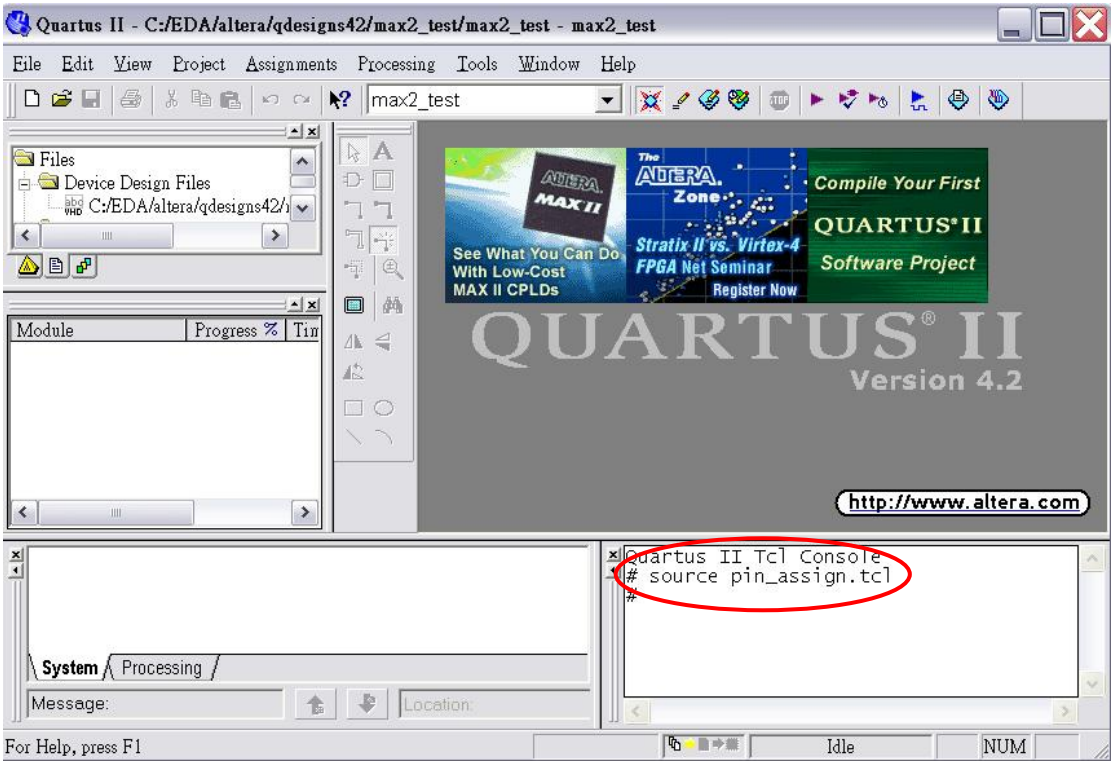
步驟8. 按【Finish】完成新增專案

步驟9. 點選【View】→【Utility Windows】→【Tcl Console】，開啟 Tcl Console 設定腳位

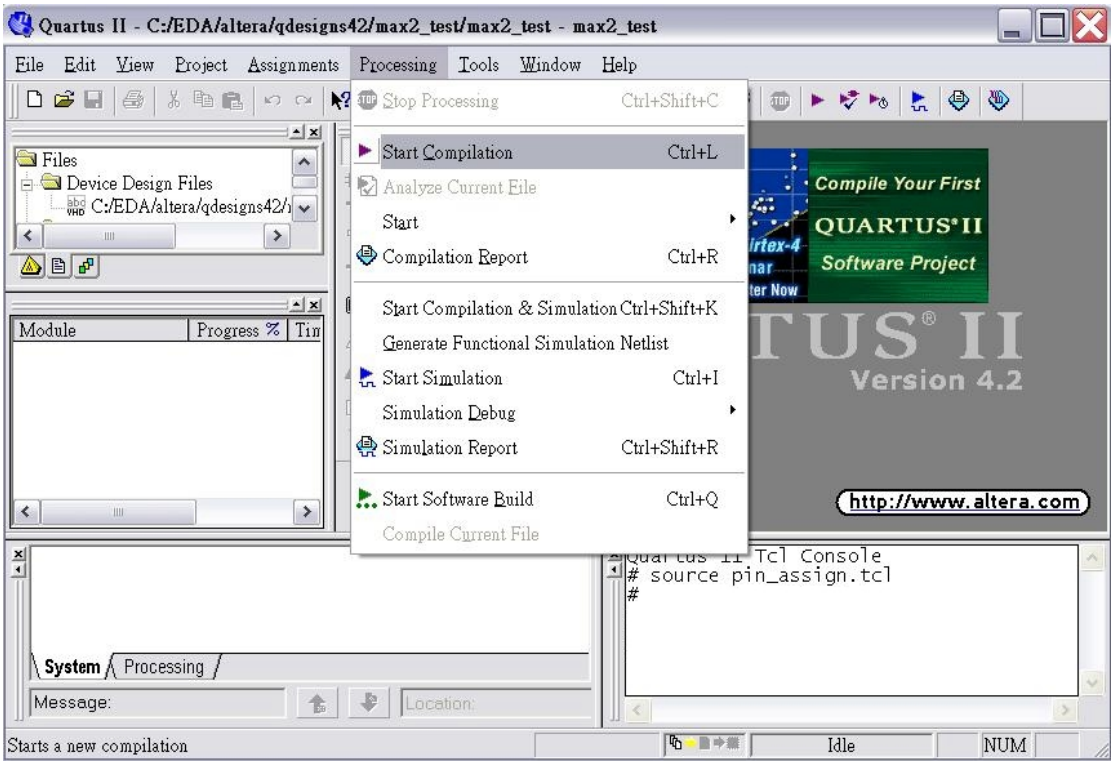


步驟10. 在 Tcl Console 視窗下輸入下列指令完成腳位設定

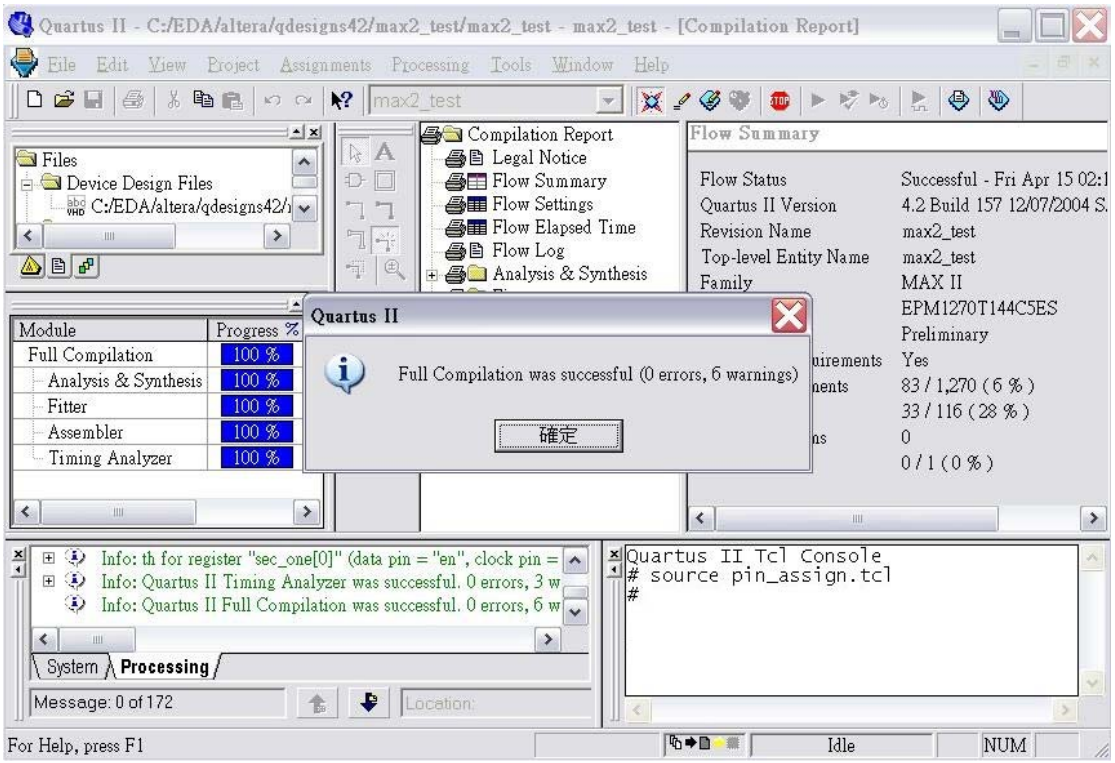
source pin_assign.tcl



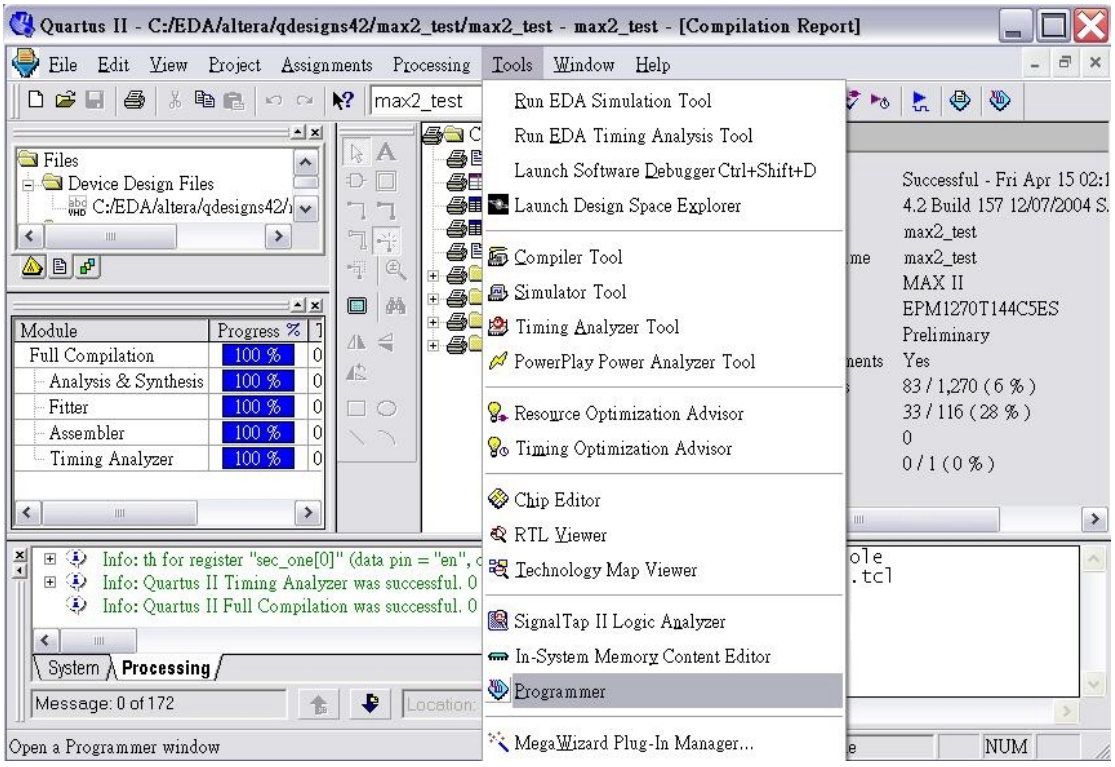
步驟11. 點選【Processing】→【Start Compilation】



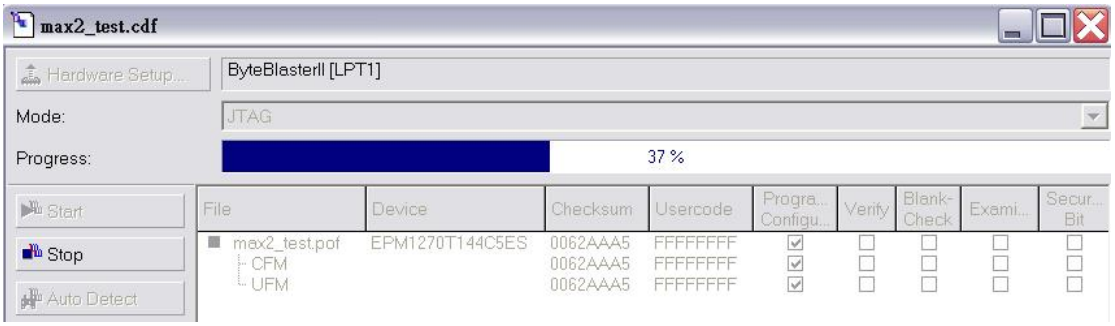
步驟12. 完成編譯


Quartus II - C:/EDA/altera/qdesigns42/max2_test/max2_test - max2_test - [Compilation Report]
File Edit View Project Assignments Processing Tools Window Help
max2_test
Compilation Report
Legal Notice
Flow Summary
Flow Settings
Flow Elapsed Time
Flow Log
Analysis & Synthesis
Flow Summary
Flow Status Successful - Fri Apr 15 02:1
Quartus II Version 4.2 Build 157 12/07/2004 S.
Revision Name max2_test
Top-level Entity Name max2_test
Family MAX II
EPM1270T144C5ES
Preliminary
Yes
83 / 1,270 (6 %)
33 / 116 (28 %)
0
0 / 1 (0 %)
Module Progress
Full Compilation 100 %
Analysis & Synthesis 100 %
Fitter 100 %
Assembler 100 %
Timing Analyzer 100 %
Quartus II
Full Compilation was successful (0 errors, 6 warnings)
確定
Info: th for register "sec_one[0]" (data pin = "en", clock pin =
Info: Quartus II Timing Analyzer was successful. 0 errors, 3 w
Info: Quartus II Full Compilation was successful. 0 errors, 6 w
Quartus II Tcl Console
source pin_assign.tcl
#
System Processing
Message: 0 of 172
Location:
For Help, press F1
Idle
NUM

步驟13. 點選【Tools】→【Programmer】



步驟14. Hardware Setup 選擇 Download cable，選擇 max2_test.pof 下載檔，勾選 Program/Configure，按下【Start】開始燒錄



MAX II 實驗板測試程式碼與說明

【MAX II 實驗板測試程式碼】

```
1  ----------------------------------------
2  --範例名稱：MAX2 Starter Kit 測試
3  --功    能：DIP ON -> LED 亮
4  --          7Segment:時鐘功能;顯示分與秒
5  --          SW1:重置;SW2:制能
6  --          SW3:關閉分時間;SW4:關閉秒時間
7  ----------------------------------------
8  library ieee;
9  use ieee.std_logic_1164.all;
10 use ieee.std_logic_unsigned.all;
11 use ieee.std_logic_arith.all;
12
13 entity max2_test is
14   Port(
15        clk,rst,en: in std_logic;
16        dip      : in std_logic_vector(7 downto 0);
17        sw3,sw4   : in std_logic;
18        led      : out std_logic_vector(7 downto 0);
19        seven_seg :out std_logic_vector(7 downto 0);
20        dig4,dig3,dig2,dig1:out std_logic);
21 end max2_test;
22
23 architecture arch of max2_test is
24   signal clk_cnt,clk_scan:std_logic;
25 signal min_ten,min_one,sec_ten,sec_one:std_logic_vector(3 downto 0);
26   signal com_sten,com_sone,com_mten,com_mone:std_logic;
27   signal en_d0,en_d1,en_d2,ce:std_logic;
28 begin
29
30 led<=dip;
31
32 dig4<=com_mten when sw3='1' else '0';
33 dig3<=com_mone when sw3='1' else '0';
34 dig2<=com_sten when sw4='1' else '0';
35 dig1<=com_sone when sw4='1' else '0';
```

```
36
37 clk_div:block
38   signal cnt  : std_logic_vector(23 downto 0);
39   signal reset: std_logic;
40 begin
41   process (clk)
42   begin
43     if reset='1' then
44       cnt<=(others=>'0');
45     elsif clk'event and clk='1' then
46       cnt<=cnt+1;
47     end if;
48   end process;
49   reset<='1' when cnt=16000000 else '0';
50   clk_cnt<=cnt(23);
51   clk_scan<=cnt(13);
52 end block clk_div;
53
54 counter:block
55 begin
56   process (clk_cnt,rst,en)
57   begin
58     if rst='0' then
59       min_ten<="0000";min_one<="0000";
60       sec_ten<="0000";sec_one<="0000";
61     elsif clk_cnt'event and clk_cnt='1' then
62       if ce='1' then
63         if sec_one="1001" then
64           if sec_ten="0101" then
65             if min_one="1001" then
66               if min_ten="0101" then
67                 min_ten<="0000";min_one<="0000";
68                 sec_ten<="0000";sec_one<="0000";
69               else
70                 min_ten<=min_ten+1;min_one<="0000";
71                 sec_ten<="0000";sec_one<="0000";
72               end if;
73             else
74               min_one<=min_one+1;
75               sec_ten<="0000";sec_one<="0000";
76             end if;
77           else
78             sec_ten<=sec_ten+1;
79             sec_one<="0000";
80           end if;
81         else
82           sec_one<=sec_one+1;
83         end if;
84       end if;
85     end if;
86   end process;
87 end block counter;
88
89 scan:block
90   signal bin: std_logic_vector(3 downto 0);
91   signal sel: integer range 0 to 3;
92 begin
93   process (clk_scan,rst)
94   begin
95     if rst='0' then
96       com_sten<='0';com_sone<='0';com_mten<='0';com_mone<='0';
```

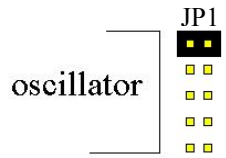
```
97      sel<=0;
98    elsif clk_scan'event and clk_scan='1' then
99      sel<=sel+1;
100      case sel is
101        when 0 =>
102          bin<=sec_one;
103          com_mten<='0';com_mone<='0';com_sten<='0';com_sone<='1';
104        when 1 =>
105          bin<=sec_ten;
106          com_mten<='0';com_mone<='0';com_sten<='1';com_sone<='0';
107        when 2 =>
108          bin<=min_one;
109          com_mten<='0';com_mone<='1';com_sten<='0';com_sone<='0';
110        when 3 =>
111          bin<=min_ten;
112          com_mten<='1';com_mone<='0';com_sten<='0';com_sone<='0';
113        when others =>
114          null;
115      end case;
116    end if;
117  end process;
118
119  process (bin)
120  begin
121    case bin is
122      when "0000" => seven_seg <= "01000000";  -- 0
123      when "0001" => seven_seg <= "01111001";  -- 1
124      when "0010" => seven_seg <= "00100100";  -- 2
125      when "0011" => seven_seg <= "00110000";  -- 3
126      when "0100" => seven_seg <= "00011001";  -- 4
127      when "0101" => seven_seg <= "00010010";  -- 5
128      when "0110" => seven_seg <= "00000010";  -- 6
129      when "0111" => seven_seg <= "01111000";  -- 7
130      when "1000" => seven_seg <= "00000000";  -- 8
131      when "1001" => seven_seg <= "00010000";  -- 9
132      when others => seven_seg <= "01111111";
133    end case;
134  end process;
135 end block scan;
136
137  process (clk)
138  begin
139    if clk'event and clk='1' then
140      en_d1<=en_d0;en_d0<=en;
141      if en_d2='1' then
142        ce<=not ce;
143      end if;
144    end if;
145  end process;
146  en_d2<= not en_d0 and en_d1;
147 end arch;
```

◎MAX II 實驗板測試程式碼說明

行　號	說　明

1~7	註解說明
8~11	元件庫宣告
13~21	輸入與輸出埠宣告
23~28	內部訊號宣告
30	dip 輸入信號指定給 led 輸出
32~33	SW3 爲 high 時七段顯示器 Dig3 與 Dig4 正常顯示，low 時不顯示
34~35	SW4 爲 high 時七段顯示器 Dig1 與 Dig2 正常顯示，low 時不顯示
37~52	除頻電路，產生 clk_cnt 計數時脈與 clk_scan 七段顯示器掃描時脈
54 ~ 87	時鐘計數器
89~117	七段顯示器多工掃描電路
119~135	七段顯示器解碼電路
137~147	致能信號轉態電路，其中第 140 行爲致能信號延遲電路，第 146 行爲致能信號下緣微分電路

硬體設定與操作說明

硬體介面	說明
	➢ Jumper 插上 JP1 的第一組提供系統頻率至 MAX II Pin 18
	➢ 指撥開關為 ON 時相對應的 LED 會亮，OFF 時相對應的 LED 會滅
	➢ 可由指撥開關控制 LED 的亮滅
	➢ 彈跳按鍵可控制七段顯示器狀態 ➢ SW1 為重置按鍵，按下時七段顯示器不顯示，放開後由 0000 開始計時 ➢ SW2 為致能鍵，按一下時七段顯示器暫停計時，再按一下時七段顯示器繼續計時 ➢ SW3 為 Dig3 與 Dig4 的致能鍵，按下時 Dig3 與 Dig4 不亮 ➢ SW4 為 Dig1 與 Dig2 的致能鍵，按下時 Dig1 與 Dig2 不亮
	➢ 顯示時鐘的分與秒功能，左邊兩個七段顯示器(Dig4, Dig3)為顯示分的功能，右邊兩個七段顯示器(Dig2, Dig1)為顯示秒的功能

第四章　ALU 範例

功能說明

此範例目的在於實現一個能將兩個輸入值執行邏輯和算術運算功能的 ALU 電路架構，其功能模式由 M、S1 和 S0 來選擇，M 的作用是作算術或邏輯運算設定，當 M=0 時執行邏輯運算，M=1 則執行算術運算，另外 S1、S0 則決定其運算的模式，各種邏輯及算術運算各有四種功能。相關功能如下表所示：

ALU 功能表

M	S1	S0	Operation
0	0	0	F<=A and B
0	0	1	F<=A or B
0	1	0	F<=A xor B
0	1	1	F<=A nand B
1	0	0	F<=A+B
1	0	1	F<=A-B
1	1	0	F<=A+1
1	1	1	F<=A-1

ALU 程式碼與說明

【ALU 程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity alu is
7  port (
8        M  :in  std_logic;
9        S  :in  std_logic_vector(1 downto 0);
10       A,B:in  std_logic_vector(2 downto 0);
11       F  :out std_logic_vector(2 downto 0));
12 end alu;
13
14 architecture a of alu is
15 begin
16   process(S,A,B)
```

```
17   begin
18   if M='0' then
19      case S is
20        when "00"=>   --"000"
21           F<=A and B;
22        when "01"=>   --"001"
23           F<=A or B;
24        when "10"=>   --"010"
25           F<=A xor B;
26        when others=> --"011"
27           F<=A nand B;
28      end case;
29    else
30      case S is
31        when "00"=>   --"100"
32           F<=A+B;
33        when "01"=>   --"101"
34           F<=A-B;
35        when "10"=>   --"110"
36           F<=A + "001";
37        when others=> --"111"
38           F<=A - "001";
39      end case;
40     end if;
41  end process;
42 end a;
```

◎ALU 程式碼說明

行　號	說　明
1~4	元件庫宣告
6~12	輸入與輸出埠宣告
18~28	邏輯運算
29~39	算術運算

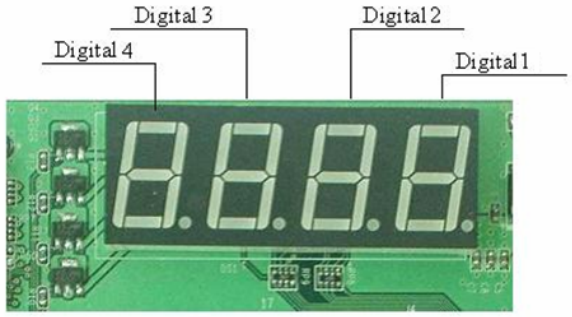
硬體設定與操作說明

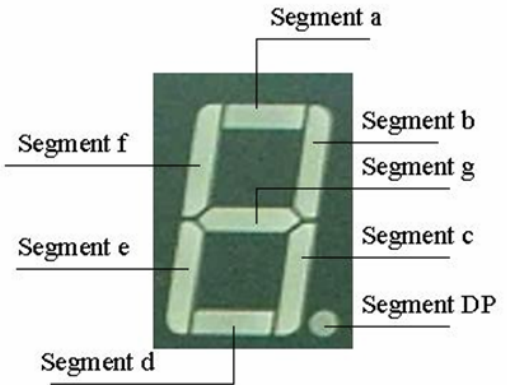
硬體介面	說明
	➢ SW1 => M
	➢ 1=>A(2) ; 2=>A(1) ; 3=>A(0) ➢ 4=>B(2) ; 5=>B(1) ; 6=>B(0) ➢ 7=> S0 ; 8=>S1
	➢ LED1=>F(2) ; LED2=>F(1) ; LED3=>F(0)
備註	需注意按下 SW 時為低電位'0'，指撥開關 ON 時為低電位'0'，LED 為低電位'0'時會亮

第五章　七段顯示器範例

功能說明

在 MAX II 實驗板上是採用四合一的七段顯示器元件作為數字輸出介面，而其共同點是共陽極且為高電位驅動，當 a,b,…,dp 等輸入是低電位時，其各自對應的 LED 燈（七段顯示器字節）就會被點亮。當 Digital 1 接高電位時，最右邊的數字會被點亮，而若 Digital 4 接高電位時，最左邊的數字會被點亮。由於四合一七段顯示器元件上每個七段顯示器上的相同字節都已被連接在一起，所以當 Digital 1, Digital 2, Digital 3, Digital 4 都接上高電位時，資料會同時傳送到所有的七段顯示器上而形成四個數字都被點亮的情況。本範例的功能為藉由指撥開關的 Dip5, Dip6, Dip7, Dip8 來做為二進制數字的輸入，以彈跳按鍵 SW1, SW2, SW3, SW4 作為 Digital 4, Digital 3, Digital 2, Digital 1 的信號控制，當按下 SW1 時 Digital 4 為高電位並點亮最左邊的數字，顯示的數字會依照指撥開關 Dip5, Dip6, Dip7, Dip8 的二進制編碼轉換為十六進制編碼輸出顯示。





ALU 程式碼與說明

【七段顯示器程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity seven_seg is
```

```
7    port (
8          sw     : in  std_logic_vector(4 downto 1);
9          dip    : in  std_logic_vector(5 to 8);
10         seg_out: out std_logic_vector(7 downto 0);
11         d      : out std_logic_vector(1 to 4)
12        );
13 end seven_seg;
14
15 architecture arch of seven_seg is
16 begin
17   d<= not sw;
18   process (dip)
19   begin
20     case dip is
21       when "0000" => seg_out <= "01000000";  -- 0   active low '0'
22       when "0001" => seg_out <= "01111001";  -- 1
23       when "0010" => seg_out <= "00100100";  -- 2
24       when "0011" => seg_out <= "00110000";  -- 3
25       when "0100" => seg_out <= "00011001";  -- 4
26       when "0101" => seg_out <= "00010010";  -- 5
27       when "0110" => seg_out <= "00000010";  -- 6
28       when "0111" => seg_out <= "01111000";  -- 7
29       when "1000" => seg_out <= "00000000";  -- 8
30       when "1001" => seg_out <= "00010000";  -- 9
31       when "1010" => seg_out <= "00001000";  -- a
32       when "1011" => seg_out <= "00000011";  -- b
33       when "1100" => seg_out <= "01000110";  -- c
34       when "1101" => seg_out <= "00100001";  -- d
35       when "1110" => seg_out <= "00000110";  -- e
36       when "1111" => seg_out <= "00001110";  -- f
37       when others => seg_out <= "11111111";
38     end case;
39   end process;
40 end arch;
```

◎七段顯示器程式碼說明

行　號	說　明
1~4	元件庫宣告
6~13	輸入與輸出埠宣告
17	彈跳按鍵的反向信號輸出給七段顯示器共同點
18~39	將指撥開關的二進制碼解碼爲十六進制的七段顯示器數字輸出

硬體設定與操作說明

硬體介面	說明
	➢ 右邊四個指撥開關為二進制編碼輸入，Dip5 為最高位元，Dip8 為最低位元，需注意指撥開關 ON 時為低電位'0'
	➢ 彈跳按鍵可控制七段顯示器狀態 ➢ 按下 SW1 時，最左邊的七段顯示器會顯示 ➢ 按下 SW2 時，左邊第二個七段顯示器會顯示 ➢ 按下 SW3 時，左邊第三個七段顯示器會顯示 ➢ 按下 SW4 時，最右邊的七段顯示器會顯示
	➢ 依照指撥開關的二進制輸入與彈跳按鍵的共同點致能來顯示十六進制數字

第六章　時鐘範例

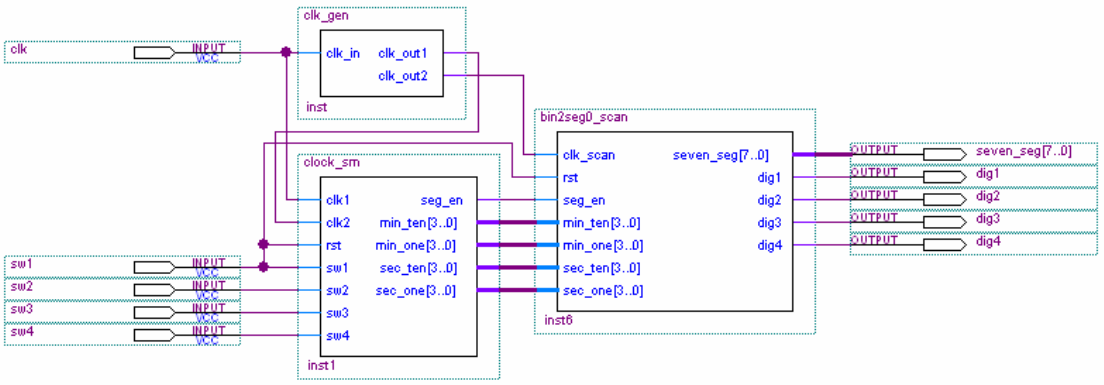
功能說明

本範例利用除頻電路，七段顯示器掃描電路，與狀態機程式設計技巧完成一個時鐘的範例，主要使用了 MAX II 實驗板上的四個彈跳按鍵與四位數七段顯示器，本範例的功能如下：

- 可在四位數七段顯示器上顯示時鐘的分與秒功能
- 具有重置、開始、暫停與設定時間功能
- 可分別針對四個七段顯示器設定數字
- 按下 SW1 時，時間會重置並顯示 0000
- 按下 SW2 時，七段顯示器會開始計時，範圍可從 00 分 00 秒計時到 59 分 59 秒
- 按下 SW3 時，可個別設定七段顯示器的數字，由最左邊的數字開始設定且其他七段顯示器會顯示 F 並保留原本的數字資料，每按一次 SW3，則可設定下一個七段顯示器的數字
- 當在設定時間的情形按下 SW4，則數字會加一
- 當時鐘開始計時的時候按下 SW4，則時間會暫停，需再按一次 SW2 才會繼續計時

時鐘程式碼與說明

下圖為時鐘範例的電路圖(clock.bdf)



【除頻電路程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity clk_gen is
6    port(
7         clk_in  : in  std_logic;
8         clk_out1: out std_logic;
9         clk_out2: out std_logic
10       );
11 end clk_gen;
12
13 architecture a of clk_gen is
14   signal cnt  : std_logic_vector(23 downto 0);
15   signal reset: std_logic;
16 begin
17   process (clk_in)
18   begin
19     if reset='1' then
20       cnt<=(others=>'0');
21     elsif clk_in'event and clk_in='1' then
22       cnt<=cnt+1;
23     end if;
24   end process;
25   reset<='1' when cnt=16000000 else '0';
26   clk_out1<=cnt(23);
27   clk_out2<=cnt(13);
28 end a;
```

◎除頻電路程式碼說明

行　號	說　明
1~3	元件庫宣告
5~11	輸入與輸出埠宣告
13~16	內部訊號宣告
19~25	上數計數器
26	時鐘計數時脈
27	七段顯示器掃描時脈

【時鐘控制狀態機程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  entity clock_sm is
7    port(
8        clk1    : in  std_logic;
9        clk2    : in  std_logic;
10       rst     : in  std_logic;
11       sw1     : in  std_logic;
12       sw2     : in  std_logic;
13       sw3     : in  std_logic;
14       sw4     : in  std_logic;
15       seg_en  : out std_logic;
16       min_ten :out std_logic_vector(3 downto 0);
17       min_one :out std_logic_vector(3 downto 0);
18       sec_ten :out std_logic_vector(3 downto 0);
19       sec_one :out std_logic_vector(3 downto 0)
20      );
21 end clock_sm;
22
23 architecture arch of clock_sm is
24  type state_type is (reset,pause,start,set_d1,set_d2,set_d3,set_d4);
25  signal present_state,next_state : state_type;
26  signal cnt_min_one,cnt_min_ten:std_logic_vector(3 downto 0);
27  signal cnt_sec_one,cnt_sec_ten:std_logic_vector(3 downto 0);
28  signal sw3_d0,sw3_d1,sw3_d2: std_logic;
29  signal sw4_d0,sw4_d1,sw4_d2: std_logic;
30  signal clk2_d0,clk2_d1,clk2_d2: std_logic;
31 begin
32   seg_en<='1';
33   sw3_d2<=sw3_d1 and not sw3_d0;
34   sw4_d2<=sw4_d1 and not sw4_d0;
35   clk2_d2<=not clk2_d1 and clk2_d0;
36
37   process(present_state)
38   begin
39     case present_state is
40       when reset  =>if    sw2='0' then next_state<=start;
41                   elsif sw3_d2='1' then next_state<=set_d1;
42                   else  next_state<=reset;
43                   end if;
44       when pause  =>if    sw2='0' then next_state<=start;
45                   elsif sw3_d2='1' then next_state<=set_d1;
46                   elsif sw1='0' then next_state<=reset;
47                   else  next_state<=pause;
48                   end if;
49       when start => if    sw3_d2='1' then next_state<=set_d1;
50                   elsif sw4='0' then next_state<=pause;
51                   elsif sw1='0' then next_state<=reset;
52                   else  next_state<=start;
53                   end if;
54       when set_d1 =>if    sw2='0' then next_state<=start;
55                   elsif sw3_d2='1' then next_state<=set_d2;
56                   else  next_state<=set_d1;
57                   end if;
58       when set_d2 =>if    sw2='0' then next_state<=start;
59                   elsif sw3_d2='1' then next_state<=set_d3;
60                   else  next_state<=set_d2;
61                   end if;
```

```
62      when set_d3 =>if    sw2='0' then next_state<=start;
63                  elsif sw3_d2='1' then next_state<=set_d4;
64                  else  next_state<=set_d3;
65                  end if;
66      when set_d4 =>if    sw2='0' then next_state<=start;
67                  elsif sw3_d2='1' then next_state<=set_d1;
68                  else  next_state<=set_d4;
69                  end if;
70    end case;
71  end process;
72
73  process(clk1,rst)
74  begin
75    if rst='0' then
76      present_state<=reset;
77    elsif clk1'event and clk1='1' then
78      present_state<=next_state;
79      sw3_d1<=sw3_d0;sw3_d0<=sw3;
80      sw4_d1<=sw4_d0;sw4_d0<=sw4;
81      clk2_d1<=clk2_d0;clk2_d0<=clk2;
82    end if;
83  end process;
84
85  process(clk1)
86  begin
87    if clk1'event and clk1='1' then
88      case present_state is
89        when reset =>
90          min_ten<="0000";min_one<="0000";
91          sec_ten<="0000";sec_one<="0000";
92          cnt_min_ten<="0000"; cnt_min_one<="0000";
93          cnt_sec_ten<="0000"; cnt_sec_one<="0000";
94        when pause  =>
95          min_ten<=cnt_min_ten;min_one<=cnt_min_one;
96          sec_ten<=cnt_sec_ten;sec_one<=cnt_sec_one;
97        when start =>
98          if clk2_d2='1' then
99            if cnt_sec_one="1001" then
100             if cnt_sec_ten="0101" then
101               if cnt_min_one="1001" then
102                 if cnt_min_ten="0101" then
103                   cnt_min_ten<="0000";cnt_min_one<="0000";
104                   cnt_sec_ten<="0000";cnt_sec_one<="0000";
105                 else
106                   cnt_min_ten<=cnt_min_ten+1;cnt_min_one<="0000";
107                   cnt_sec_ten<="0000";cnt_sec_one<="0000";
108                 end if;
109               else
110                 cnt_min_one<=cnt_min_one+1;
111                 cnt_sec_ten<="0000";cnt_sec_one<="0000";
112               end if;
113             else
114               cnt_sec_ten<=cnt_sec_ten+1;
115               cnt_sec_one<="0000";
116             end if;
117           else
118             cnt_sec_one<=cnt_sec_one+1;
119           end if;
120         end if;
121         min_ten<=cnt_min_ten;min_one<=cnt_min_one;
122         sec_ten<=cnt_sec_ten;sec_one<=cnt_sec_one;
```

```
123         when set_d1=>
124           if sw4_d2='1' then
125             if cnt_min_ten = 5 then cnt_min_ten <= "0000";
126             else cnt_min_ten <= cnt_min_ten + 1;
127             end if;
128           end if;
129           min_ten<=cnt_min_ten;min_one<="1111";
130           sec_ten<="1111";sec_one<="1111";
131         when set_d2=>
132           if sw4_d2='1' then
133             if cnt_min_one = 9 then cnt_min_one <= "0000";
134             else cnt_min_one <= cnt_min_one + 1;
135             end if;
136           end if;
137           min_ten<="1111";min_one<=cnt_min_one;
138           sec_ten<="1111";sec_one<="1111";
139         when set_d3=>
140           if sw4_d2='1' then
141             if cnt_sec_ten = 5 then cnt_sec_ten <= "0000";
142             else cnt_sec_ten <= cnt_sec_ten + 1;
143             end if;
144           end if;
145           min_ten<="1111";min_one<="1111";
146           sec_ten<=cnt_sec_ten;sec_one<="1111";
147         when set_d4=>
148           if sw4_d2='1' then
149             if cnt_sec_one = 9 then cnt_sec_one <= "0000";
150             else cnt_sec_one <= cnt_sec_one + 1;
151             end if;
152           end if;
153           min_ten<="1111";min_one<="1111";
154           sec_ten<="1111";sec_one<=cnt_sec_one;
155       end case;
156     end if;
157   end process;
158 end arch;
```

◎時鐘控制狀態機程式碼說明

行　號	說　明
1~4	元件庫宣告
6~21	輸入與輸出埠宣告
23~31	內部訊號宣告
32	致能七段顯示器掃描電路
33~35	SW3, SW4, CLK2 微分電路，功能是偵測輸入信號是否變化

37~71	時鐘狀態機的組合邏輯電路
73~83	時鐘狀態機的順序邏輯電路與 SW3,SW4,CLK2 的延遲電路
85~157	時鐘狀態機的輸出解碼電路

【多工掃描七段顯示器程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity bin2seg0_scan is
7    port (
8         clk_scan  : in  std_logic;
9         rst       : in  std_logic;
10        seg_en    : in  std_logic;
11        min_ten   : in  std_logic_vector(3 downto 0);
12        min_one   : in  std_logic_vector(3 downto 0);
13        sec_ten   : in  std_logic_vector(3 downto 0);
14        sec_one   : in  std_logic_vector(3 downto 0);
15        seven_seg : out std_logic_vector(7 downto 0);
16        dig1,dig2,dig3,dig4: out std_logic
17       );
18 end bin2seg0_scan;
19
20 architecture arch of bin2seg0_scan is
21   signal bin: std_logic_vector(3 downto 0);
22   signal sel: integer range 0 to 3;
23 begin
24   process (clk_scan,rst)
25   begin
26     if rst='0' then
27       dig2<='0';dig1<='0';dig4<='0';dig3<='0';
28       sel<=0;
29     elsif clk_scan'event and clk_scan='1' then
30       sel<=sel+1;
31       case sel is
32         when 0 =>
33           bin<=sec_one;
34           dig4<='0';dig3<='0';dig2<='0';dig1<='1';
35         when 1 =>
36           bin<=sec_ten;
37           dig4<='0';dig3<='0';dig2<='1';dig1<='0';
38         when 2 =>
39           bin<=min_one;
40           dig4<='0';dig3<='1';dig2<='0';dig1<='0';
41         when 3 =>
42           bin<=min_ten;
43           dig4<='1';dig3<='0';dig2<='0';dig1<='0';
44         when others =>
45           null;
46       end case;
47     end if;
48   end process;
```

```
49
50   process (bin)
51   begin
52     case bin is
53       when "0000" => seven_seg <= "01000000";
54       when "0001" => seven_seg <= "01111001";
55       when "0010" => seven_seg <= "00100100";
56       when "0011" => seven_seg <= "00110000";
57       when "0100" => seven_seg <= "00011001";
58       when "0101" => seven_seg <= "00010010";
59       when "0110" => seven_seg <= "00000010";
60       when "0111" => seven_seg <= "01111000";
61       when "1000" => seven_seg <= "00000000";
62       when "1001" => seven_seg <= "00010000";
63       when "1111" => seven_seg <= "00001110";
64       when others => seven_seg <= "11111111";
65     end case;
66   end process;
67 end arch;
```

◎多工掃描七段顯示器程式碼說明

行　號	說　明
1~4	元件庫宣告
6~18	輸入與輸出埠宣告
20~23	內部訊號宣告
24~48	七段顯示器共同點掃描電路
50~66	七段顯示器數字輸出解碼電路

硬體設定與操作說明

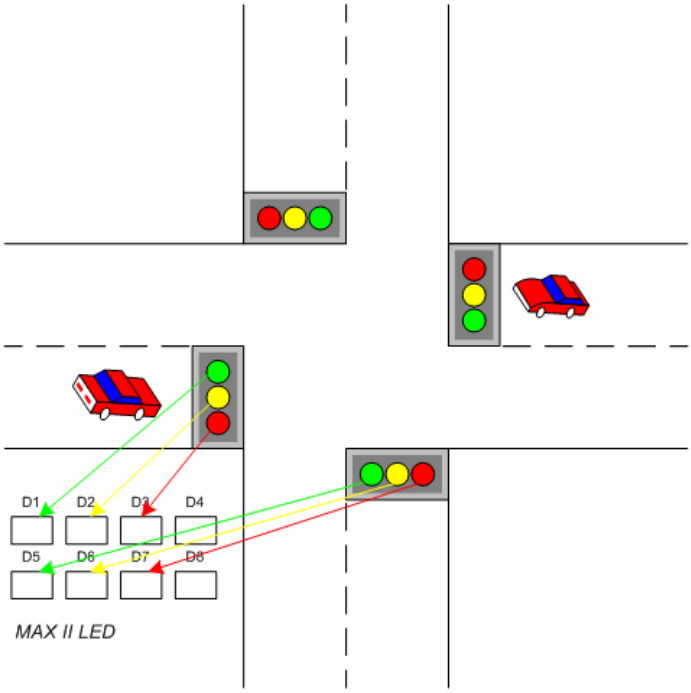
硬體介面	說明
	➢ 時鐘功能的控制鍵 ➢ SW1 為時鐘重置鍵，按下後時間會重置成為 00 分 00 秒 ➢ SW2 為開始計時鍵，當重置後與設定時間後需要按下 SW2 讓時鐘開始計時 ➢ SW3 為時間設定鍵，當按下 SW3 時可以開始設定七段顯示器的數字，由最左邊的數字開始設定，其他三個數字會顯示 F，每按一次 SW4 則數字加一，每按一次 SW3 則可以設定下一個七段顯示器數字 ➢ SW4 為暫停或數字加一鍵，當時鐘在計時過程中按下 SW4，則時間會暫停，當時鐘在設定時間模式時按下 SW4，則數字會加一
	➢ 依照彈跳按鍵的設定來顯示時鐘的分與秒時間

第七章　紅綠燈範例

功能說明

本範例設計一個具有重置和手動控制功能的十字路口紅綠燈控制，以 SW1 作為重置按鈕，指撥開關可控制紅綠燈模式為正常、維持東西向道路通行、維持南北向道路通行等模式。東西向道路之綠、黃、紅燈分別顯示在 D1, D2, D3 LED，南北向道路之綠、黃、紅燈分別顯示在 D5, D6, D7 LED。而且當東西向道路通行或南北向道路通行時，在七段顯示器的 Digital 4 或 Digital 1 會顯示倒數計時的時間。

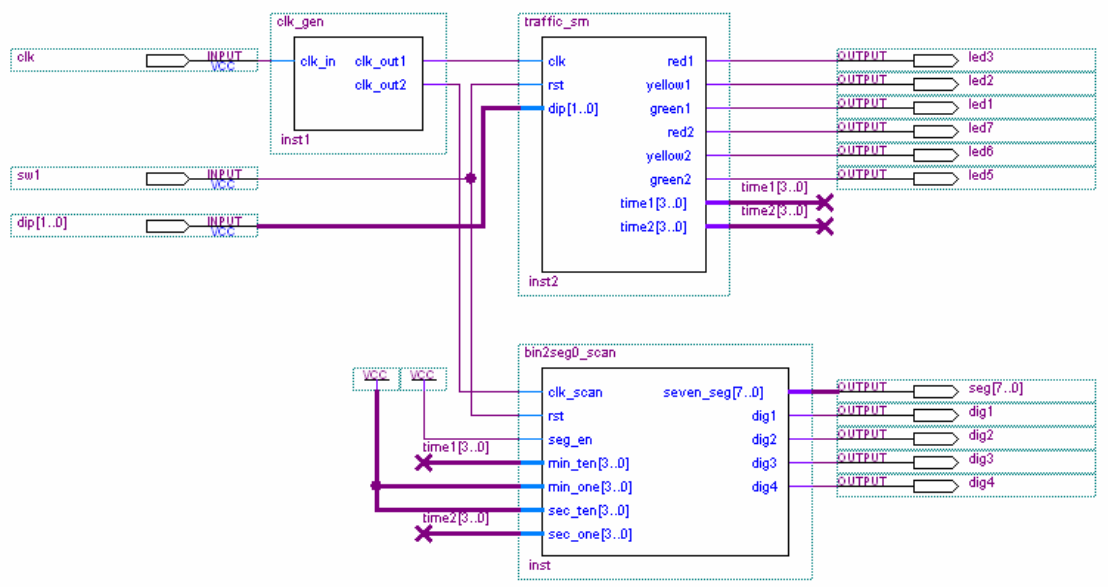
當觸動 Reset 鍵(SW1)時，系統將被重置，路口燈號為東西向道路綠燈(D1)亮、南北向道路紅燈(D7)亮，東西向道路開始倒數計時可通行時間並顯示在七段顯示器 Digital 4 上。當南北向道路的綠燈亮起時，南北向道路開始倒數計時可通行時間並顯示在七段顯示器 Digital 1 上。若要手動控制維持東西向道路通行的話則可設定開關為 01(DIP2 ON, DIP1 OFF)，此時燈號變化至東西向道路的綠燈亮起之後便能維持狀態不變，直到開關改變才能再發生燈號的變動。另外，若要維持南北向道路通行的話則可設定開關為 10(DIP2 OFF, DIP1 ON)。



十字路口紅綠燈

紅綠燈程式碼與說明

下圖為紅綠燈電路圖(traffic.bdf)，其中 CLK_GEN 模組為除頻電路，提供計數時脈與七段顯示器掃描時脈。TRAFFIC_SM 為紅綠燈控制狀態機電路。BIN2SEG0_SCAN 為七段顯示器掃描電路。



紅綠燈電路圖(traffic.bdf)

【除頻電路程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity clk_gen is
6    port(
7         clk_in  : in  std_logic;
8         clk_out1: out std_logic;
9         clk_out2: out std_logic
10      );
11 end clk_gen;
12
13 architecture a of clk_gen is
14   signal cnt  : std_logic_vector(23 downto 0);
15   signal reset: std_logic;
16 begin
17   process (clk_in)
18   begin
19     if reset='1' then
20       cnt<=(others=>'0');
```

```
21    elsif clk_in'event and clk_in='1' then
22      cnt<=cnt+1;
23    end if;
24  end process;
25  reset<='1' when cnt=16000000 else '0';
26  clk_out1<=cnt(23);
27  clk_out2<=cnt(13);
28 end a;
```

◎除頻電路程式碼說明

行　號	說　明
1~3	元件庫宣告
5~11	輸入與輸出埠宣告
13~16	內部訊號宣告
17~25	上數計數器
26	時鐘計數時脈
27	七段顯示器掃描時脈

【紅綠燈控制狀態機程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity traffic_sm is
5    port (
6         clk : in std_logic;
7         rst : in std_logic;
8         dip : in std_logic_vector(1 downto 0);
9       red1, yellow1, green1 : out std_logic;
10      red2, yellow2, green2 : out std_logic;
11      time1,time2: out std_logic_vector(3 downto 0)
12     );
13 end traffic_sm;
14
15 architecture a of traffic_sm is
16   type state is (s0,s1,s2,s3,s4,s5,s6,s7);
17   signal present_state,next_state : state;
18 begin
19   process (present_state, dip)
20   begin
21     red1 <= '1'; yellow1 <= '1'; green1 <= '1';
22     red2 <= '1'; yellow2 <= '1'; green2 <= '1';
23     time1<= "1111"; time2<="1111";
```

```
24    case present_state is
25   when s0 =>
26       green1 <= '0';  red2 <= '0';
27       time1<="0011";
28       if dip= "01" then
29           next_state <= s0;
30         else
31           next_state <= s1;
32         end if;
33     when s1 =>
34         green1 <= '0';  red2 <= '0';
35         time1<="0010";
36         next_state <= s2;
37     when s2 =>
38         green1 <= '0';  red2 <= '0';
39         time1<="0001";
40         next_state <= s3;
41     when s3 =>
42         yellow1 <= '0'; red2 <= '0';
43         time1<="0000";
44         next_state <= s4;
45     when s4 =>
46         red1 <= '0';    green2 <= '0';
47         time2<="0011";
48         if dip= "10" then
49           next_state <= s4;
50         else
51           next_state <= s5;
52         end if;
53     when s5 =>
54         red1 <= '0';    green2 <= '0';
55         time2<="0010";
56         next_state <= s6;
57     when s6 =>
58         red1 <= '0';    green2 <= '0';
59         time2<="0001";
60         next_state <= s7;
61     when s7 =>
62         red1 <= '0';    yellow2 <= '0';
63         time2<="0000";
64         next_state <= s0;
65  end case;
66   end process;
67
68   process(rst,clk)
69   begin
70     if (rst='0') then
71       present_state <= s0;
72     elsif clk'event and clk='1' then
73       present_state <= next_state ;
74     end if ;
75   end process;
76 end a;
```

◎紅綠燈控制狀態機程式碼說明

行　號	說　明

1~2	元件庫宣告
4~13	輸入與輸出埠宣告
15~18	內部訊號宣告
19~66	紅綠燈狀態機的組合邏輯電路
68~75	紅綠燈狀態機的順序邏輯電路

【七段顯示器掃描電路程式碼】

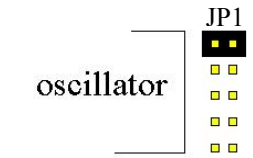
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity bin2seg0_scan is
7    port (
8          clk_scan  : in  std_logic;
9          rst       : in  std_logic;
10         seg_en    : in  std_logic;
11         min_ten   : in  std_logic_vector(3 downto 0);
12         min_one   : in  std_logic_vector(3 downto 0);
13         sec_ten   : in  std_logic_vector(3 downto 0);
14         sec_one   : in  std_logic_vector(3 downto 0);
15         seven_seg : out std_logic_vector(7 downto 0);
16         dig1,dig2,dig3,dig4: out std_logic
17        );
18 end bin2seg0_scan;
19
20 architecture arch of bin2seg0_scan is
21   signal bin: std_logic_vector(3 downto 0);
22   signal sel: integer range 0 to 3;
23 begin
24   process (clk_scan,rst)
25   begin
26     if rst='0' then
27       dig2<='0';dig1<='0';dig4<='0';dig3<='0';
28       sel<=0;
29     elsif clk_scan'event and clk_scan='1' then
30       sel<=sel+1;
31       case sel is
32         when 0 =>
33           bin<=sec_one;
34           dig4<='0';dig3<='0';dig2<='0';dig1<='1';
35         when 1 =>
36           bin<=sec_ten;
37           dig4<='0';dig3<='0';dig2<='1';dig1<='0';
38         when 2 =>
39           bin<=min_one;
40           dig4<='0';dig3<='1';dig2<='0';dig1<='0';
41         when 3 =>
```

```
42          bin<=min_ten;
43          dig4<='1';dig3<='0';dig2<='0';dig1<='0';
44        when others =>
45          null;
46      end case;
47    end if;
48  end process;
49
50  process (bin)
51  begin
52    case bin is
53      when "0000" => seven_seg <= "01000000";
54      when "0001" => seven_seg <= "01111001";
55      when "0010" => seven_seg <= "00100100";
56      when "0011" => seven_seg <= "00110000";
57      when "0100" => seven_seg <= "00011001";
58      when "0101" => seven_seg <= "00010010";
59      when "0110" => seven_seg <= "00000010";
60      when "0111" => seven_seg <= "01111000";
61      when "1000" => seven_seg <= "00000000";
62      when "1001" => seven_seg <= "00010000";
63      when others => seven_seg <= "11111111";
64    end case;
65  end process;
66 end arch;
```

◎七段顯示器掃描電路程式碼說明

行　號	說　明
1~4	元件庫宣告
6~18	輸入與輸出埠宣告
20~23	內部訊號宣告
24~48	七段顯示器共同點掃描電路
50~65	七段顯示器數字輸出解碼電路

硬體設定與操作說明

硬體介面	說明
	➢ Jumper 插上 JP1 的第一組提供系統頻率至 MAX II Pin 18
	➢ SW1 為重置鍵
	➢ 控制開關(DIP2,DIP1) ➢ DIP1→OFF, DIP2→ON ： 維持東西向道路通行，D1(東西向綠燈)與 D7(南北向紅燈)亮 ➢ DIP1→ON, DIP2→OFF ： 維持南北向道路通行，D3(東西向紅燈)與 D5(南北向綠燈)亮 ➢ DIP1→OFF, DIP2→OFF 或 DIP1→ON, DIP2→ON : 紅綠燈正常運作，東西向道路通行或南北向道路通行時在七段顯示器的 Digital 4 或 Digital 1 會顯示倒數計時的時間
	➢ 東西向道路之綠、黃、紅燈(D1,D2,D3) ➢ 南北向道路之綠、黃、紅燈(D5,D6,D7)
	➢ 顯示綠燈剩餘時間 ➢ Digital 4 顯示東西向道路綠燈剩餘時間 ➢ Digital 1 顯示南北向道路綠燈剩餘時間

第八章　遊戲機範例

功能說明

本範例是一個可以兩個人玩比大小的遊戲機，以 SW1 作為重新開始的按鍵，SW2 為玩家 A 的決定鍵，SW3 為玩家 B 的決定鍵。遊戲的順序如下。

步驟1. 指撥開關 DIP1 是設定哪位玩家的數字較大時為勝利，若 DIP1 OFF 則當玩家 A 的數字大於玩家 B 的數字時勝利，若 DIP1 ON 則當玩家 B 的數字大於玩家 A 的數字時勝利，勝利時 LED 會全部同時閃爍，沒有勝利時 LED 會全滅

步驟2. 按下 SW1 重置遊戲機，此時 LED 燈會以繞圈圈的方式不停的亮

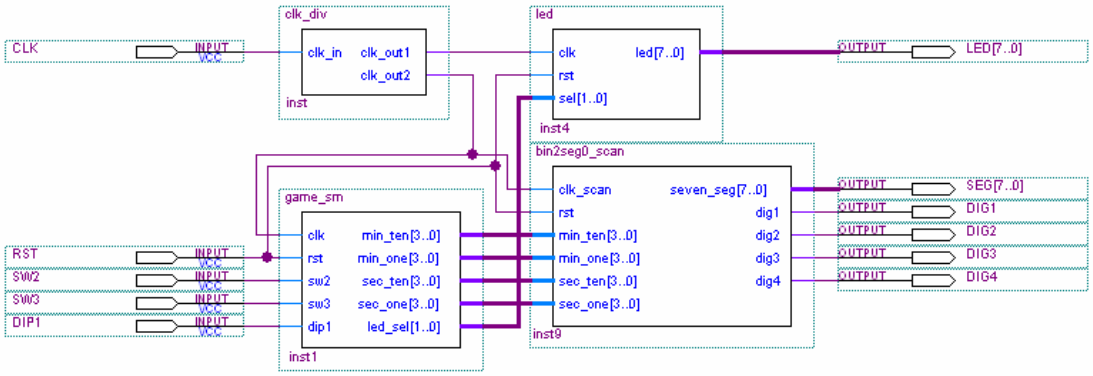
步驟3. 玩家 A 按下 SW2，七段顯示器的左邊顯示出一組亂數產生的數字

步驟4. 玩家 B 按下 SW3，七段顯示器的右邊顯示出一組亂數產生的數字

步驟5. 步驟 1 為設定勝利條件，重複步驟 2~4 的動作重新玩遊戲

遊戲機程式碼與說明

下圖為遊戲機電路圖(game.bdf)，其中 CLK_DIV 模組為除頻電路，提供計數時脈與七段顯示器掃描時脈。GAME_SM 為遊戲機控制狀態機電路。BIN2SEG0_SCAN 為七段顯示器掃描電路。LED 模組為 LED 顯示電路。



遊戲機電路圖(game.bdf)

【除頻電路程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity clk_div is
6    port(
7         clk_in  : in  std_logic;
8         clk_out1: out std_logic;
9         clk_out2: out std_logic
10      );
11 end clk_div;
12
13 architecture a of clk_div is
14   signal cnt  : std_logic_vector(23 downto 0);
15   signal reset: std_logic;
16 begin
17   process (clk_in)
18   begin
19     if reset='1' then
20       cnt<=(others=>'0');
21     elsif clk_in'event and clk_in='1' then
22       cnt<=cnt+1;
23     end if;
24   end process;
25   reset<='1' when cnt=16000000 else '0';
26   clk_out1<=cnt(19);
27   clk_out2<=cnt(14);
28 end a;
```

◎除頻電路程式碼說明

行　號	說　明
1~3	元件庫宣告
5~11	輸入與輸出埠宣告
13~16	內部訊號宣告
17~25	上數計數器
26	LED 計數時脈
27	七段顯示器掃描與亂數產生器時脈

【遊戲機控制狀態機程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  entity game_sm is
7    port(
8         clk    : in  std_logic;
9         rst    : in  std_logic;
10        sw2,sw3 : in  std_logic;
11        dip1   : in  std_logic;
12        min_ten : out std_logic_vector(3 downto 0);
13        min_one : out std_logic_vector(3 downto 0);
14        sec_ten : out std_logic_vector(3 downto 0);
15        sec_one : out std_logic_vector(3 downto 0);
16        led_sel : out std_logic_vector(1 downto 0)
17      );
18 end game_sm;
19
20 architecture a of game_sm is
21   type state_type is (S0,S1,S2);
22   signal present_state,next_state : state_type;
23   signal cnta1,cnta2,cntb1,cntb2 : std_logic_vector(3 downto 0);
24   signal randoma,randomb:std_logic;
25 begin
26   min_ten<=cnta2;min_one<=cnta1;
27   sec_ten<=cntb2;sec_one<=cntb1;
28
29   process(present_state,rst,sw2,sw3,dip1)
30   begin
31     case present_state is
32       when S0 =>
33             randoma<='1';randomb<='1';
34             led_sel<="01";
35             if sw2='0' then
36               next_state<=S1;
37             else
38               next_state<=S0;
39             end if;
40       when S1 =>
41             randoma<='0';randomb<='1';
42             led_sel<="01";
43             if sw3='0' then
44               next_state<=S2;
45             else
46               next_state<=S1;
47             end if;
48       when S2 =>
49             randoma<='0';randomb<='0';
50             if rst='0' then
51               next_state<=S0;
52             else
53               next_state<=S2;
54             end if;
55             if dip1='1' then
56               if cnta2>cntb2 then
57                 led_sel<="10";
58               elsif cnta2=cntb2 and cnta1>cntb1 then
59                 led_sel<="10";
60               else
```

```
61                led_sel<="00";
62              end if;
63            else
64              if cntb2>cnta2 then
65                led_sel<="10";
66              elsif cntb2=cnta2 and cntb1>cnta1 then
67                led_sel<="10";
68              else
69                led_sel<="00";
70              end if;
71            end if;
72    end case;
73  end process;
74
75  process(clk,rst)
76  begin
77    if rst='0' then
78      present_state<=S0;
79    elsif clk'event and clk='1' then
80      present_state<=next_state;
81    end if;
82  end process;
83
84  process(clk,randoma)
85  begin
86    if clk'event and clk='1' then
87      if randoma='1' then
88        if cnta2 = "1001" then
89          if cnta1 = "1001" then
90            cnta1 <= "0000";
91            cnta2 <= "0000";
92          else
93            cnta1 <= cnta1 + 1;
94          end if;
95        elsif cnta1 = "1001" then
96          cnta1 <= "0000";
97          cnta2 <= cnta2 + 1;
98        else
99          cnta1 <= cnta1 + 1;
100       end if;
101     end if;
102   end if;
103 end process;
104
105 process(clk,randomb)
106 begin
107   if clk'event and clk='1' then
108     if randomb='1' then
109       if cntb2 = "1001" then
110         if cntb1 = "1001" then
111           cntb1 <= "0000";
112           cntb2 <= "0000";
113         else
114           cntb1 <= cntb1 + 1;
115         end if;
116       elsif cntb1 = "1001" then
117         cntb1 <= "0000";
118         cntb2 <= cntb2 + 1;
119       else
120         cntb1 <= cntb1 + 1;
121       end if;
```

```
122      end if;
123    end if;
124   end process;
125 end a;
```

◎遊戲機控制狀態機程式碼說明

行　號	說　明
1~4	元件庫宣告
6~18	輸入與輸出埠宣告
20~25	內部訊號宣告
26~27	四個七段顯示器數字輸出
29~73	遊戲機狀態機的組合邏輯電路
75~82	遊戲機狀態機的順序邏輯電路
84~124	遊戲機計數器

【七段顯示器掃描電路程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity bin2seg0_scan is
7    port (
8         clk_scan  : in  std_logic;
9         rst       : in  std_logic;
10        min_ten   : in  std_logic_vector(3 downto 0);
11        min_one   : in  std_logic_vector(3 downto 0);
12        sec_ten   : in  std_logic_vector(3 downto 0);
13        sec_one   : in  std_logic_vector(3 downto 0);
14        seven_seg : out std_logic_vector(7 downto 0);
15        dig1,dig2,dig3,dig4: out std_logic
16       );
17 end bin2seg0_scan;
18
19 architecture arch of bin2seg0_scan is
20   signal bin: std_logic_vector(3 downto 0);
21   signal sel: integer range 0 to 3;
22 begin
```

```
23   process (clk_scan,rst)
24   begin
25     if rst='0' then
26       dig2<='0';dig1<='0';dig4<='0';dig3<='0';
27       sel<=0;
28     elsif clk_scan'event and clk_scan='1' then
29       sel<=sel+1;
30       case sel is
31         when 0 =>
32           bin<=sec_one;
33           dig4<='0';dig3<='0';dig2<='0';dig1<='1';
34         when 1 =>
35           bin<=sec_ten;
36           dig4<='0';dig3<='0';dig2<='1';dig1<='0';
37         when 2 =>
38           bin<=min_one;
39           dig4<='0';dig3<='1';dig2<='0';dig1<='0';
40         when 3 =>
41           bin<=min_ten;
42           dig4<='1';dig3<='0';dig2<='0';dig1<='0';
43         when others =>
44           null;
45       end case;
46     end if;
47   end process;
48
49   process (bin)
50   begin
51     case bin is
52       when "0000" => seven_seg <= "01000000";
53       when "0001" => seven_seg <= "01111001";
54       when "0010" => seven_seg <= "00100100";
55       when "0011" => seven_seg <= "00110000";
56       when "0100" => seven_seg <= "00011001";
57       when "0101" => seven_seg <= "00010010";
58       when "0110" => seven_seg <= "00000010";
59       when "0111" => seven_seg <= "01111000";
60       when "1000" => seven_seg <= "00000000";
61       when "1001" => seven_seg <= "00010000";
62       when "1111" => seven_seg <= "00001110";
63       when others => seven_seg <= "11111111";
64     end case;
65   end process;
66 end arch;
```

◎七段顯示器掃描電路程式碼說明

行　號	說　明
1~4	元件庫宣告
6~17	輸入與輸出埠宣告

19~22	內部訊號宣告
23~47	七段顯示器共同點掃描電路
49~65	七段顯示器數字輸出解碼電路

【LED 顯示電路程式碼】

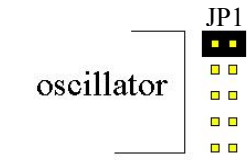
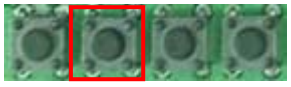
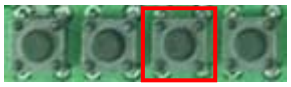
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity led is
7    port(
8        clk :in  std_logic;
9        rst :in  std_logic;
10       sel :in  std_logic_vector(1 downto 0);
11       led :out std_logic_vector(7 downto 0)
12      );
13 end led;
14
15 architecture a of led  is
16   signal cnt   : std_logic_vector(2 downto 0);
17   signal led_t1,led_t2:std_logic_vector(7 downto 0);
18 begin
19
20   process(clk,rst)
21   begin
22     if rst='0' then
23       cnt<="000";
24     elsif clk'event and clk='1' then
25        cnt<=cnt+1;
26     end if;
27   end process;
28
29   led_t1<="11111110" when cnt="000" else
30          "11111101" when cnt="001" else
31          "11111011" when cnt="010" else
32          "11110111" when cnt="011" else
33          "01111111" when cnt="100" else
34          "10111111" when cnt="101" else
35          "11011111" when cnt="110" else
36          "11101111";
37   led_t2<="00000000" when cnt="000" else
38          "11111111" when cnt="001" else
39          "00000000" when cnt="010" else
40          "11111111" when cnt="011" else
41          "00000000" when cnt="100" else
42          "11111111" when cnt="101" else
43          "00000000" when cnt="110" else
44          "11111111";
45
46   led <= led_t1 when sel="01" else
```

```
47         led_t2 when sel="10" else
48         "11111111";
49 end a;
```

◎LED 顯示電路程式碼說明

行　號	說　明
1~4	元件庫宣告
6~13	輸入與輸出埠宣告
15~18	內部訊號宣告
20~27	上數計數器
29~44	LED 解碼電路
46~48	LED 多工輸出電路

硬體設定與操作說明

硬體介面	說明
	➢ Jumper 插上 JP1 的第一組提供系統頻率至 MAX II Pin 18
	➢ SW1 為重置鍵
	➢ SW2 為玩家 A 的決定鍵
	➢ SW3 為玩家 B 的決定鍵
	➢ DIP1 是設定哪位玩家的數字較大時為勝利，若 DIP1 OFF 則當玩家 A 的數字大於玩家 B 的數字時勝利，若 DIP1 ON 則當玩家 B

	的數字大於玩家 A 的數字時勝利
	➢ 重置後開始玩遊戲時 LED 會以繞圈圈的方式不停的亮 ➢ 勝利時 LED 會全部同時閃爍 ➢ 沒有勝利時 LED 會全滅
	➢ 玩家 A 的數字
	➢ 玩家 B 的數字

第九章　RS-232 範例

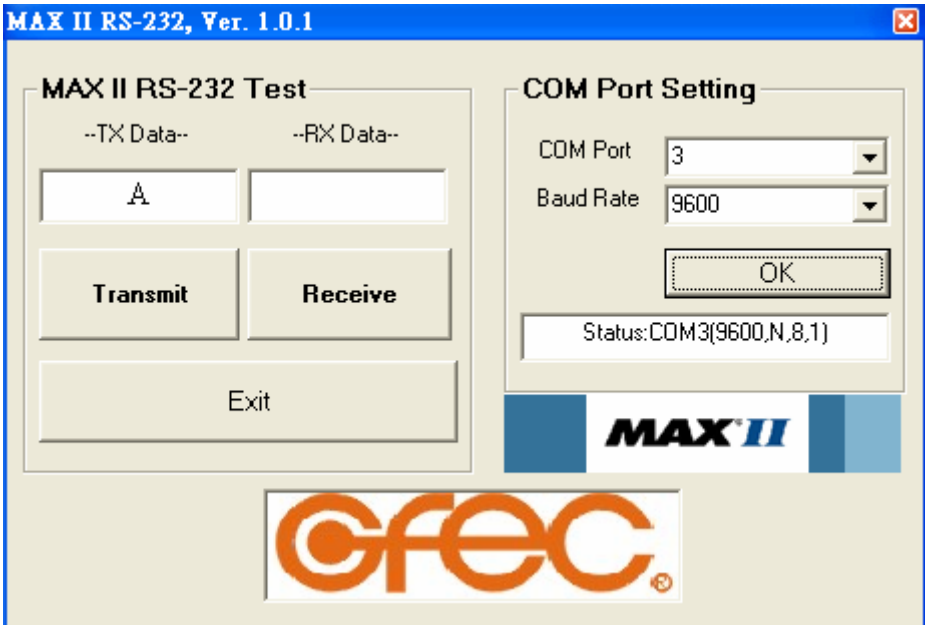
功能說明

本範例為測試 UART 串列資料傳輸與接收功能，藉由電腦端的 Comm Port 傳送串列資料給 MAX II 實驗板，第一次傳送資料給 MAX II 實驗板時，會將接收到的資料顯示在七段顯示器的 Digital 1 與 Digital 2，並將資料存放在內部暫存器中，當有下一筆資料傳送給 MAX II 實驗板時，上一筆資料會被移位顯示在七段顯示器的 Digital 3 與 Digital 4，新的資料會顯示在七段顯示器的 Digital 1 與 Digital 2，並且更新內部暫存器資料。此外也可以由 MAX II 實驗板將內部暫存器的資料以串列的方式透過 Comm Port 回傳給電腦端並顯示在螢幕上。以下為本範例的測試步驟：

步驟1.　以 9 Pin 的 RS-232 Cable 連接 MAX II 實驗板與電腦的 Comm Port

步驟2.　下載 uart.pof 檔至 MAX II 實驗板中

步驟3.　執行 MAX II RS-232.exe 應用程式，外觀如下圖所示



步驟4.　在應用程式中設定所使用的 COM Port 與鮑率，選擇好後按下【OK】，使用者必須在電腦中確認所使用的 COM Port 號碼，在 MAX II 實驗板上的指撥開關 DIP1, DIP2, DIP3 設定好鮑率

步驟5. 在 TX Data 中輸入一字元，按下【Transmit】

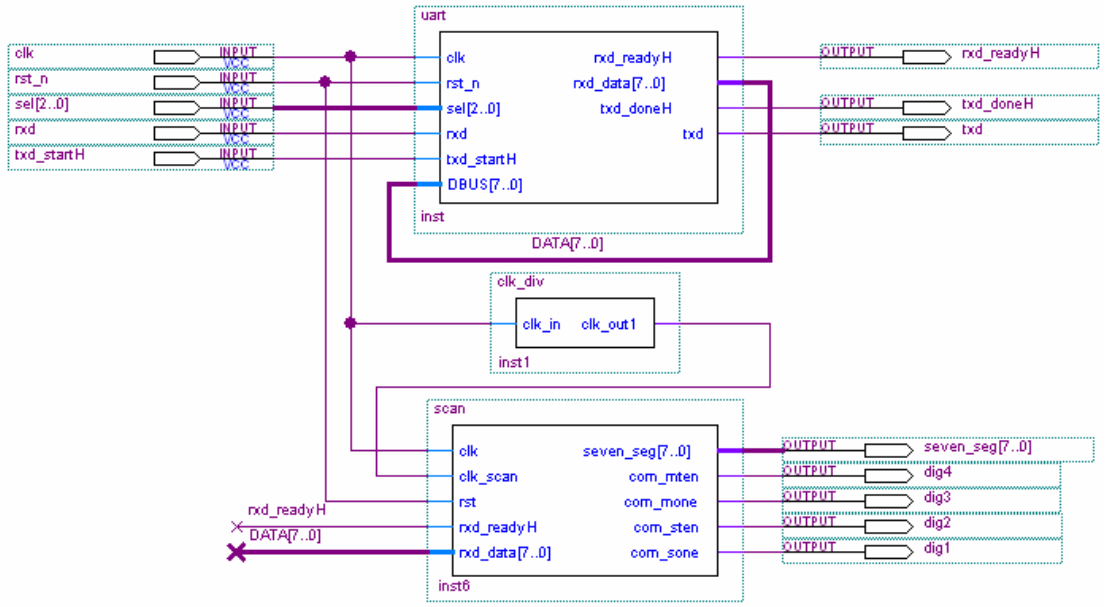
步驟6. 此時在 MAXII 實驗板上的七段顯示器會顯示電腦端所傳送字元的 ASCII 碼

步驟7. 按下 MAXII 實驗板上的 SW2，UART 所接收的 ASCII 碼回傳給電腦端

步驟8. 按下【Receive】

步驟9. 在接收區將顯示出 ASCII 碼所代表的字元

以下為 MAXII 中的 UART 功能規格



信號腳	I/O 位置		功能說明
CLK	GCLK0P	P18	系統工作頻率
RST_N	SW1	P23	系統重置信號
SEL[2:0]	DIP3,DIP2,DIP1	P16,P21,P22	鮑率選擇
RXD	RS-232_RXD	P49	RS-232 接收信號
TXD_STARTH	SW2	P24	傳送鍵
TXD	RS-232_TXD	P45	RS-232 傳送信號
SEVEN_SEG[7..0]	SEGMENT[DIP~A]	P44~37	7 段顯示器訊號
DIG4	COMMON4	P32	Digital 4 common pin
DIG3	COMMON3	P31	Digital 3 common pin
DIG2	COMMON2	P30	Digital 2 common pin
DIG1	COMMON1	P29	Digital 1 common pin

【鮑率產生器程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity br_gen is
6    generic(divisor: integer := 26);
7  port(
8       sysclk:in std_logic;
9       sel   :in std_logic_vector(2 downto 0);
10      bclkx8:buffer std_logic;
11      bclk  :out std_logic
12     );
13 end br_gen;
14
15 architecture arch of br_gen is
16   signal cnt2,clkdiv: std_logic;
17   signal ctr2: std_logic_vector (7 downto 0):= "00000000";
18   signal ctr3: std_logic_vector (2 downto 0):= "000";
19 begin
20   process (sysclk)
21     variable cnt1    : integer range 0 to divisor;
22     variable divisor2 : integer range 0 to divisor;
23   begin
24     divisor2 := divisor/2;
25     if (sysclk'event and sysclk='1') then
26       if cnt1=divisor then
27         cnt1 := 1;
28       else
29         cnt1 := cnt1 + 1;
30       end if;
31     end if;
32     if (sysclk'event and sysclk='1') then
33       if (( cnt1=divisor2) or (cnt1=divisor)) then
34         cnt2 <= not cnt2;
35       end if;
36     end if;
37   end process;
38   clkdiv<=  cnt2 ;
39   process (clkdiv)
40   begin
41     if(rising_edge(clkdiv)) then
42       ctr2 <= ctr2+1;
43     end if;
44   end process;
45   bclkx8<=ctr2(CONV_INTEGER(sel));
46   process (bclkx8)
47   begin
48     if(rising_edge(bclkx8)) then
49       ctr3 <= ctr3+1;
50     end if;
51   end process;
52   bclk <= ctr3(2);
53 end arch;
```

◎鮑率產生器程式碼說明

行　號	說　明
1~3	元件庫宣告
5~13	輸入與輸出埠宣告
15~19	內部訊號宣告。
20~38	除頻器(clkdiv = sysclk / divisor)
39~44	8 位元上數計數器
45	多工器；可得到鮑率 8 倍頻率
46~52	除頻器(bclk = bclkx8 / 8)

【RS-232 接收器程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity uart_receiver is
7  port(
8        sysclk    :in  std_logic;
9        rst_n     :in  std_logic;
10       bclkx8    :in  std_logic;
11       rxd       :in  std_logic;
12       rxd_readyH:out std_logic;
13       RDR       :out std_logic_vector(7 downto 0)
14     );
15 end uart_receiver;
16
17 architecture arch of uart_receiver is
18   type statetype is (idle,start_detected,recv_data);
19   signal state,nextstate:statetype;
20   signal inc1,inc2,clr1,clr2:std_logic;
21   signal shftRSR,load_RDR:std_logic;
22   signal bclkx8_dlayed,bclkx8_rising:std_logic;
23   signal RSR:std_logic_vector(7 downto 0);
24   signal ct1:integer range 0 to 7;
25   signal ct2:integer range 0 to 8;
26   signal ok_en: std_logic;
27 begin
28 bclkx8_rising<=bclkx8 and(not bclkx8_dlayed);
29 process(state,rxd,ct1,ct2,bclkx8_rising)
30 begin
```

```
31   inc1<='0';inc2<='0';clr1<='0';clr2<='0';
32   shftRSR<='0';load_RDR<='0';ok_en<='0';
33   case state is
34     when idle=>
35       if (rxd='0') then
36         nextstate<=start_detected;
37       else
38         nextstate<=idle;
39       end if;
40     when start_detected=>
41       if (bclkx8_rising='0') then
42         nextstate<=start_detected;
43       elsif (rxd='1') then
44         clr1<='1';
45         nextstate<=idle;
46       elsif (ct1=3) then
47         clr1<='1';
48         nextstate<=recv_data;
49       else
50         inc1<='1';
51         nextstate<=start_detected;
52       end if;
53     when recv_data=>
54       if (bclkx8_rising='0') then
55         nextstate<=recv_data;
56       else
57         inc1<='1';
58         if (ct1/=7) then
59           nextstate<=recv_data;
60         elsif (ct2/=8) then
61           shftRSR<='1';
62           inc2<='1';
63           clr1<='1';
64           nextstate<=recv_data;
65         elsif (rxd='0') then
66           nextstate<=idle;
67           clr1<='1';
68           clr2<='1';
69         else
70           load_RDR<='1';
71           ok_en<='1';
72           clr1<='1';
73           clr2<='1';
74           nextstate<=idle;
75         end if;
76       end if;
77     end case;
78   end  process;
79   process(sysclk,rst_n)
80   begin
81     if (rst_n='0') then
82       state<=idle;
83       bclkx8_dlayed<='0';
84       ct1<=0;
85       ct2<=0;
86       RDR<="00101011";
87     elsif (sysclk'event and sysclk='1') then
88       state<=nextstate;
89
90       if(clr1='1')then ct1<=0;elsif(inc1='1')then ct1<=ct1+1;end if;
91
```

```
92      if(clr2='1')then ct2<=0;elsif(inc2='1')then ct2<=ct2+1;end if;
93
94      if(shftRSR='1')then RSR<=rxd & RSR(7 downto 1);end if;
95
96      if(load_RDR='1')then RDR<=RSR;end if;
97
98      if(ok_en='1')then rxd_readyH<='1';else rxd_readyH<='0';end if;
99      bclkx8_dlayed<=bclkx8;
100    end if;
101   end process;
102 end arch;
```

◎RS-232 接收器程式碼說明

行　號	說　明
1~4	元件庫宣告
6~15	輸入與輸出埠宣告
17~27	內部訊號宣告
28	bclkx8 上緣微分，偵測 bclkx8 有無上緣信號
29~78	串列傳輸接收器狀態機
79~101	串列傳輸接收器狀態與信號更新

【RS-232 傳送器程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity uart_transmitter is
5    port(
6         sysclk     : in std_logic;
7         rst_n      : in std_logic;
8         bclk       : in std_logic;
9         txd_startH : in std_logic;
10        DBUS       : in std_logic_vector(7 downto 0);
11        txd_doneH  : out std_logic;
12        txd        : out std_logic
13       );
14 end uart_transmitter;
15
16 architecture arch of uart_transmitter is
17   type statetype is (idle, synch, tdata);
18   signal state, nextstate : statetype;
```

```
19   signal tsr : std_logic_vector (8 downto 0);
20   signal bct: integer range 0 to 9;
21   signal inc, clr, loadTSR, shftTSR, start: std_logic;
22   signal bclk_rising, bclk_dlayed, txd_done: std_logic;
23   signal txd_startH_d0,txd_startH_d1,txd_startH_d2:std_logic;
24 begin
25   txd <= tsr(0);
26   txd_doneH <= txd_done;
27   bclk_rising <= bclk and (not bclk_dlayed);
28   txd_startH_d2<=not txd_startH_d0 and txd_startH_d1;
29   process(state,txd_startH_d2, bct, bclk_rising)
30   begin
31     inc <= '0';
32     clr <= '0';
33     loadTSR <= '0';
34     shftTSR <= '0';
35     start <= '0';
36     txd_done <= '0';
37  case state is
38      when idle =>
39          if (txd_startH_d2 = '1' ) then
40          loadTSR <= '1';
41            nextstate <= synch;
42        else
43            nextstate <= idle;
44          end if;
45      when synch =>
46          if (bclk_rising = '1') then
47            start <= '1';
48            nextstate <= tdata;
49          else
50            nextstate <= synch;
51          end if;
52      when tdata =>
53          if (bclk_rising = '0') then
54            nextstate <= tdata;
55          elsif (bct /= 9) then
56            shfttsr <= '1';
57            inc <= '1';
58            nextstate <= tdata;
59        else
60            clr <= '1';
61            txd_done <= '1';
62            nextstate <= idle;
63          end if;
64      end case;
65   end process;
66   process (sysclk, rst_n)
67   begin
68     if (rst_n = '0') then
69       TSR <= "111111111";
70       state <= idle;
71       bct <= 0;
72       bclk_dlayed <= '0';
73     elsif (sysclk'event and sysclk = '1') then
74       state <= nextstate;
75       txd_startH_d0<=txd_startH;txd_startH_d1<=txd_startH_d0;
76       if (clr = '1') then
77         bct <= 0;
78       elsif (inc = '1') then
79         bct <= bct + 1;
```

```
80      end if;
81      if (loadTSR = '1') then
82        TSR <= DBUS & '1';
83      elsif (start = '1') then
84        TSR(0) <= '0';
85      elsif (shftTSR = '1') then
86        TSR <= '1' & TSR(8 downto 1);
87      end if;
88      bclk_dlayed <= bclk;
89    end if;
90  end process;
91 end arch;
```

◎RS-232 傳送器程式碼說明

行　號	說　明
1~2	元件庫宣告
4~14	輸入與輸出埠宣告
16~24	內部訊號宣告
25	將 TSR 低電位傳送給 txd
27	傳送資料完畢信號
28	Bclk 正緣微分
29~65	串列傳輸傳送器狀態機
66~90	串列傳輸傳送器狀態與信號更新

【RS-232 模組程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity uart is
7  port(
8       clk      : in  std_logic;
9       rst_n    : in  std_logic;
10      sel      : in  std_logic_vector(2 downto 0);
```

```
11     rxd      : in  std_logic;
12     rxd_readyH: out std_logic;
13     rxd_data  : out std_logic_vector(7 downto 0);
14     txd_startH: in  std_logic;
15     DBUS     : in  std_logic_vector(7 downto 0);
16     txd_doneH : out std_logic;
17     txd      : out std_logic
18    );
19 end uart;
20
21 architecture a of uart is
22   component br_gen
23   generic(divisor: integer := 3);
24   port(
25       sysclk:in std_logic;
26       sel   :in std_logic_vector(2 downto 0);
27       bclkX8:buffer std_logic;
28       bclk  :out std_logic
29      );
30   end component;
31
32   component uart_receiver
33   port(
34       sysclk    :in  std_logic;
35       rst_n     :in  std_logic;
36       bclkx8    :in  std_logic;
37       rxd      :in  std_logic;
38       rxd_readyH:out std_logic;
39       RDR      :out std_logic_vector(7 downto 0)
40      );
41   end component;
42
43   component uart_transmitter
44   port(
45       sysclk    : in std_logic;
46       rst_n     : in std_logic;
47       bclk      : in std_logic;
48       txd_startH : in std_logic;
49       DBUS      : in std_logic_vector(7 downto 0);
50       txd_doneH  : out std_logic;
51       txd       : out std_logic
52      );
53   end component;
54   signal bclkx8,bclk: std_logic;
55 begin
56   u1: br_gen generic map(26)
57             port map(clk,sel,bclkx8,bclk);
58   u2:uart_receiver
59             port map(clk,rst_n,bclkx8,rxd,rxd_readyH,rxd_data);
60   u3: uart_transmitter
61             port map(clk,rst_n,bclk,txd_startH,DBUS,txd_doneH,txd);
62 end a;
```

◎RS-232 模組程式碼說明

行　號	說　明

1~4	元件庫宣告
6~19	輸入與輸出埠宣告
21~55	內部訊號宣告
56~61	模組連線

【除頻電路程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity clk_div is
6    port(
7         clk_in  : in  std_logic;
8         clk_out1: out std_logic
9       );
10 end clk_div;
11
12 architecture a of clk_div is
13   signal cnt  : std_logic_vector(23 downto 0);
14   signal reset: std_logic;
15 begin
16   process (clk_in)
17   begin
18     if reset='1' then
19       cnt<=(others=>'0');
20     elsif clk_in'event and clk_in='1' then
21       cnt<=cnt+1;
22     end if;
23   end process;
24   reset<='1' when cnt=16000000 else '0';
25   clk_out1<=cnt(13);
26 end a;
```

◎除頻電路程式碼說明

行　號	說　明
1~3	元件庫宣告
5~10	輸入與輸出埠宣告
12~15	內部訊號宣告

16~24	上數計數器
25	掃描時脈

【七段顯示器掃描電路程式碼】

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity scan is
7    port (
8          clk,clk_scan,rst : in  std_logic;
9          rxd_readyH       : in  std_logic;
10         rxd_data         : in  std_logic_vector(7 downto 0);
11         seven_seg        : out std_logic_vector(7 downto 0);
12         com_mten,com_mone: out std_logic;
13         com_sten,com_sone: out std_logic);
14 end scan;
15
16 architecture arch of scan is
17   signal bin: std_logic_vector(3 downto 0);
18   signal sel: integer range 0 to 3;
19   signal min_ten,min_one: std_logic_vector(3 downto 0);
20   signal sec_ten,sec_one: std_logic_vector(3 downto 0);
21 begin
22
23   process (clk,rxd_readyH)
24   begin
25     if rst='0' then
26        sec_one<=rxd_data(3 downto 0);
27        sec_ten<=rxd_data(7 downto 4);
28        min_one<="0000";
29        min_ten<="0000";
30     elsif clk'event and clk='1' then
31       if rxd_readyH='1' then
32         sec_one<=rxd_data(3 downto 0);
33         sec_ten<=rxd_data(7 downto 4);
34         min_one<=sec_one;
35         min_ten<=sec_ten;
36       end if;
37     end if;
38   end process;
39
40   process (clk_scan,rst)
41   begin
42     if rst='0' then
43       com_sten<='0';com_sone<='0';com_mten<='0';com_mone<='0';
44       sel<=0;
45     elsif clk_scan'event and clk_scan='1' then
46       sel<=sel+1;
47       case sel is
48         when 0 =>
49           bin<=sec_one;
50           com_mten<='0';com_mone<='0';com_sten<='0';com_sone<='1';
```

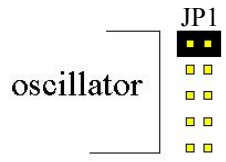
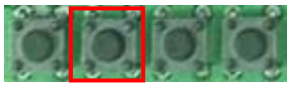
```
51        when 1 =>
52          bin<=sec_ten;
53          com_mten<='0';com_mone<='0';com_sten<='1';com_sone<='0';
54        when 2 =>
55          bin<=min_one;
56          com_mten<='0';com_mone<='1';com_sten<='0';com_sone<='0';
57        when 3 =>
58          bin<=min_ten;
59          com_mten<='1';com_mone<='0';com_sten<='0';com_sone<='0';
60        when others =>
61          null;
62      end case;
63    end if;
64  end process;
65
66  process (bin)
67  begin
68    case bin is
69      when "0000" => seven_seg <= "01000000";  -- 0
70      when "0001" => seven_seg <= "01111001";  -- 1
71      when "0010" => seven_seg <= "00100100";  -- 2
72      when "0011" => seven_seg <= "00110000";  -- 3
73      when "0100" => seven_seg <= "00011001";  -- 4
74      when "0101" => seven_seg <= "00010010";  -- 5
75      when "0110" => seven_seg <= "00000010";  -- 6
76      when "0111" => seven_seg <= "01111000";  -- 7
77      when "1000" => seven_seg <= "00000000";  -- 8
78      when "1001" => seven_seg <= "00010000";  -- 9
79      when "1010" => seven_seg <= "00001000";  -- a
80      when "1011" => seven_seg <= "00000011";  -- b
81      when "1100" => seven_seg <= "01000110";  -- c
82      when "1101" => seven_seg <= "00100001";  -- d
83      when "1110" => seven_seg <= "00000110";  -- e
84      when "1111" => seven_seg <= "00001110";  -- f
85      when others => seven_seg <= "01111111";
86    end case;
87  end process;
88 end arch;
```

◎七段顯示器掃描電路程式碼說明

行　號	說　明
1~4	元件庫宣告
6~14	輸入與輸出埠宣告
16~21	內部訊號宣告
23~38	四個七段顯示器顯示數字更新

40~64	七段顯示器共同點掃描電路
66~88	七段顯示器數字輸出解碼電路

硬體設定與操作說明

硬體介面	說明
	➢ Jumper 插上 JP1 的第一組提供系統頻率至 MAX II Pin 18
	➢ SW1 為重置鍵
	➢ SW2 為將接收到的資料傳送給電腦端
	➢ DIP1,DIP2,DIP3 為鮑率選擇開關 鮑率選擇表
	➢ 電腦端傳來字元的 ASCII 碼
	➢ 傳送給電腦端的資料

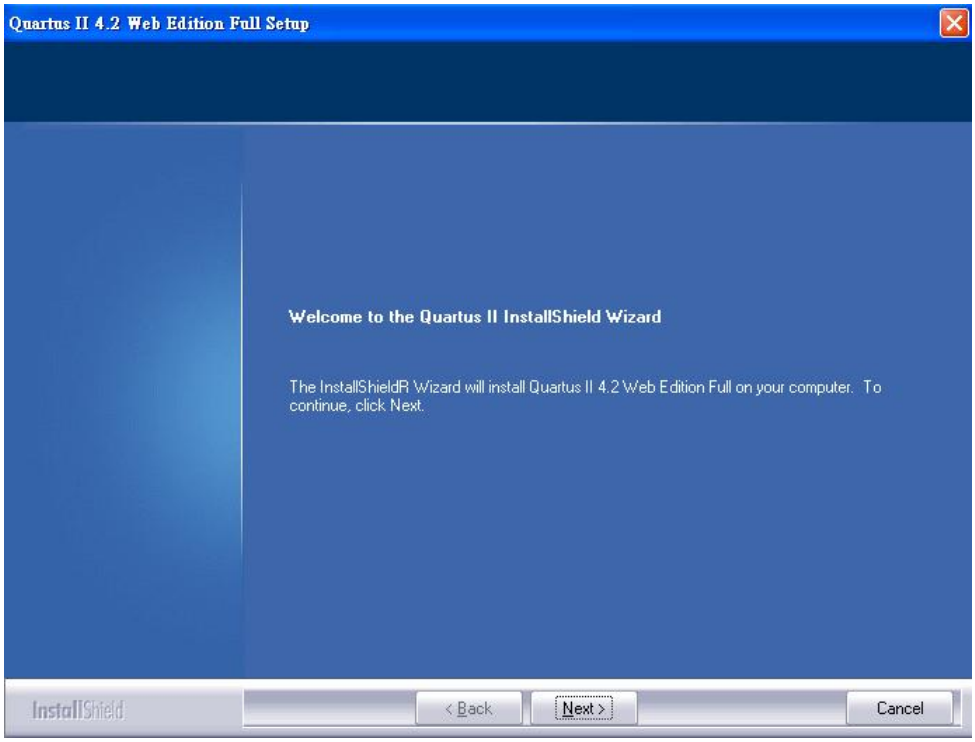
選擇信號 DIP3,DIP2,DIP1	鮑率
0 0 0	38400
0 0 1	19200
0 1 0	9600
0 1 1	4800
1 0 0	2400
1 0 1	1200
1 1 0	600
1 1 1	300

ASCII 與字元對照表

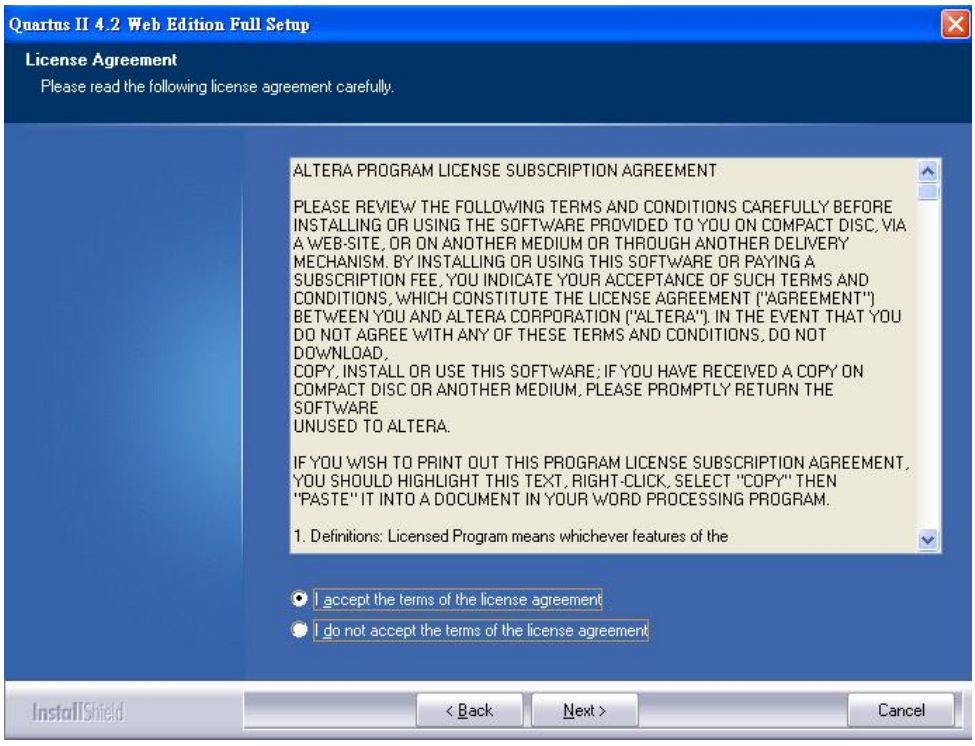
ASCII (HEX)	字元	ASCII (HEX)	字元	ASCII (HEX)	字元	ASCII (HEX)	字元	ASCII (HEX)	字元	ASCII (HEX)	字元
20	空格	30	0	40	@	50	P	60	`	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	'	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2a	*	3a	:	4a	J	5a	Z	6a	j	7a	z
2b	+	3b	;	4b	K	5b	[	6b	k	7b	{
2c	,	3c	<	4c	L	5c	\	6c	l	7c	|
2d	-	3d	=	4d	M	5d	]	6d	m	7d	}
2e	.	3e	>	4e	N	5e	^	6e	n	7e	~
2f	/	3f	?	4f	O	5f	_	6f	o	7f	DEL

附錄 A　QuartusII 4.2 安裝流程

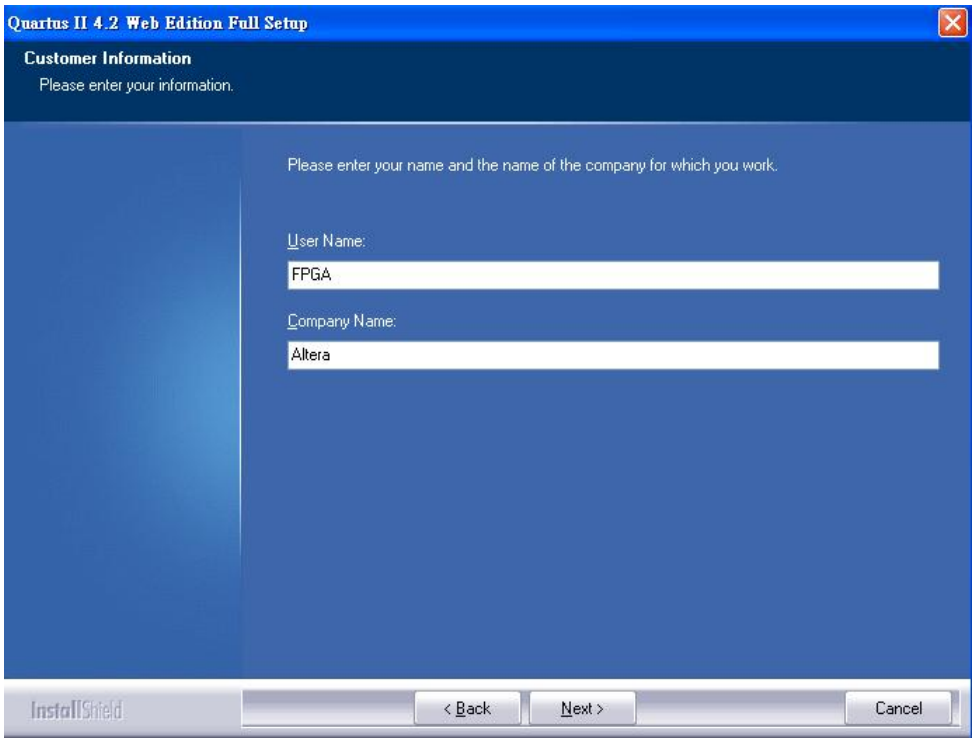
步驟1.　按【Next】


Quartus II 4.2 Web Edition Full Setup
Welcome to the Quartus II InstallShield Wizard
The InstallShieldR Wizard will install Quartus II 4.2 Web Edition Full on your computer. To continue, click Next.
InstallShield
< Back
Next >
Cancel

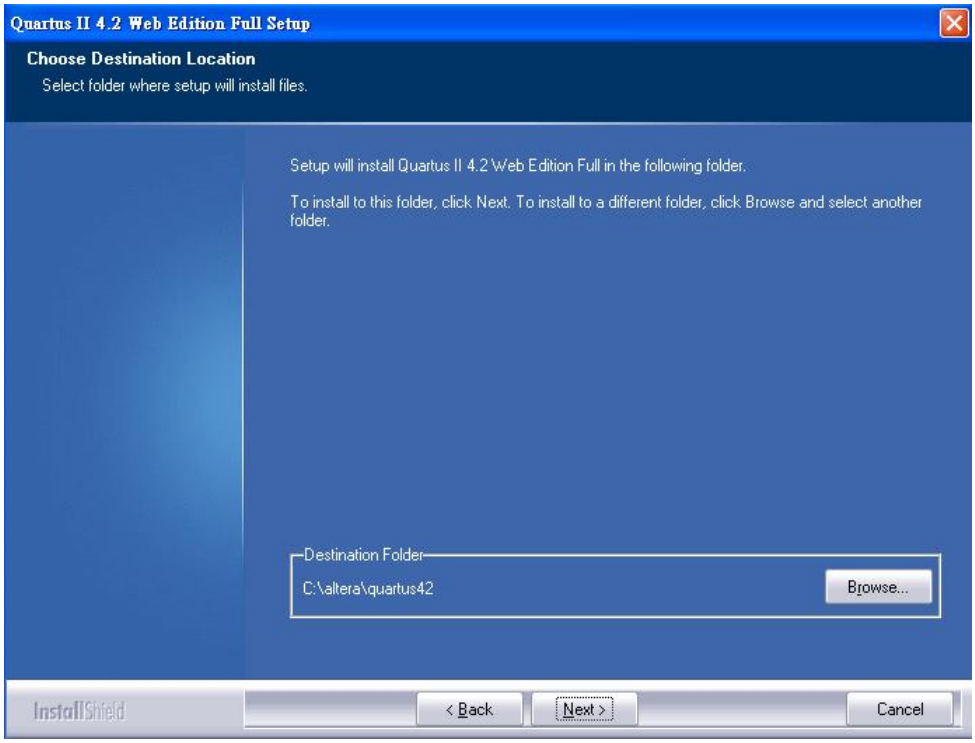
步驟2.　點選【I accept the terms of the license agreement】，按【Next】


Quartus II 4.2 Web Edition Full Setup
License Agreement
Please read the following license agreement carefully.
ALTERA PROGRAM LICENSE SUBSCRIPTION AGREEMENT
PLEASE REVIEW THE FOLLOWING TERMS AND CONDITIONS CAREFULLY BEFORE INSTALLING OR USING THE SOFTWARE PROVIDED TO YOU ON COMPACT DISC, VIA A WEB-SITE, OR ON ANOTHER MEDIUM OR THROUGH ANOTHER DELIVERY MECHANISM. BY INSTALLING OR USING THIS SOFTWARE OR PAYING A SUBSCRIPTION FEE, YOU INDICATE YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS, WHICH CONSTITUTE THE LICENSE AGREEMENT ("AGREEMENT") BETWEEN YOU AND ALTERA CORPORATION ("ALTERA"). IN THE EVENT THAT YOU DO NOT AGREE WITH ANY OF THESE TERMS AND CONDITIONS, DO NOT DOWNLOAD,
COPY, INSTALL OR USE THIS SOFTWARE; IF YOU HAVE RECEIVED A COPY ON COMPACT DISC OR ANOTHER MEDIUM, PLEASE PROMPTLY RETURN THE SOFTWARE
UNUSED TO ALTERA.
IF YOU WISH TO PRINT OUT THIS PROGRAM LICENSE SUBSCRIPTION AGREEMENT, YOU SHOULD HIGHLIGHT THIS TEXT, RIGHT-CLICK, SELECT "COPY" THEN "PASTE" IT INTO A DOCUMENT IN YOUR WORD PROCESSING PROGRAM.
1. Definitions: Licensed Program means whichever features of the
I accept the terms of the license agreement
I do not accept the terms of the license agreement
InstallShield
< Back
Next >
Cancel

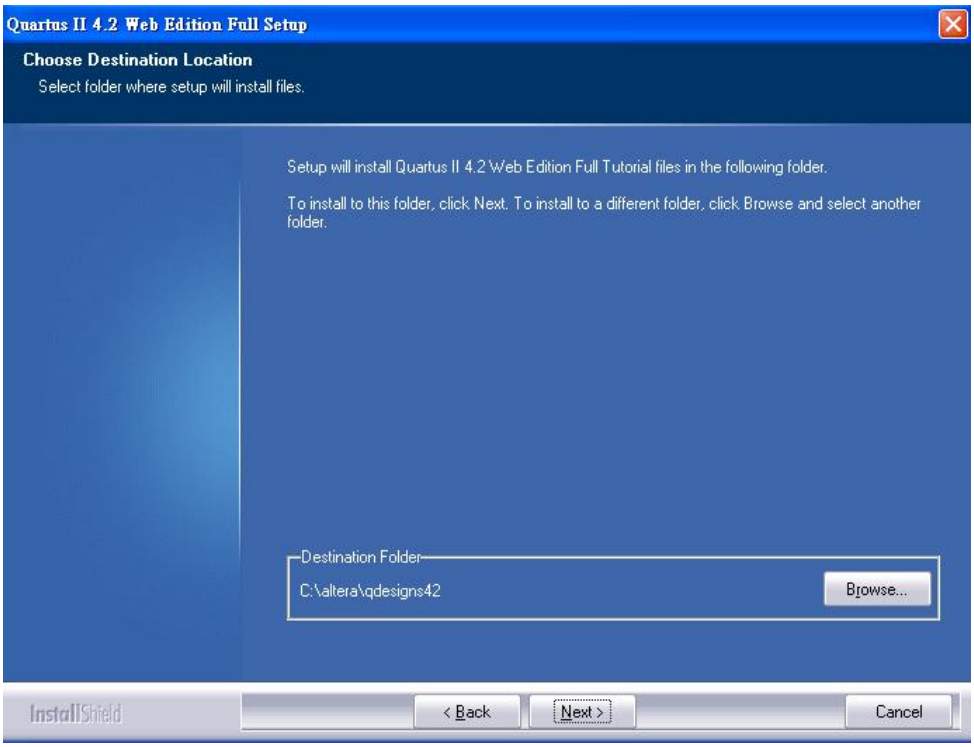
步驟3.　填寫 User Name 與 Company Name，按【Next】



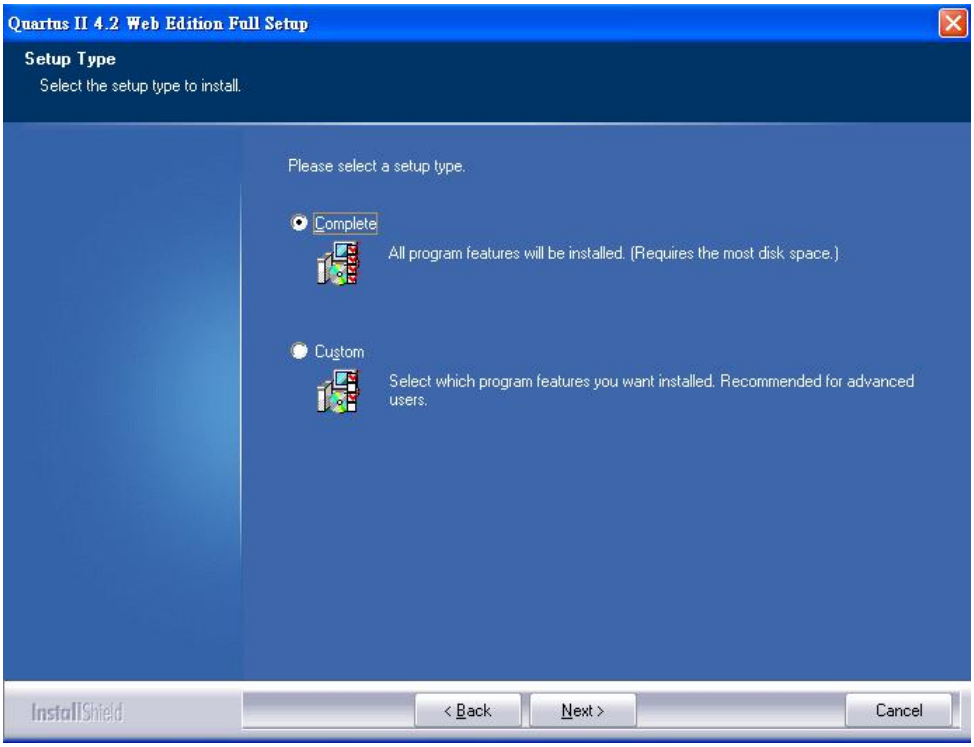
步驟4.　指定 quartus42 安裝路徑，按【Next】


Quartus II 4.2 Web Edition Full Setup
Choose Destination Location
Select folder where setup will install files.
Setup will install Quartus II 4.2 Web Edition Full in the following folder.
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.
Destination Folder
C:\altera\quartus42
Browse...
InstallShield
< Back
Next >
Cancel

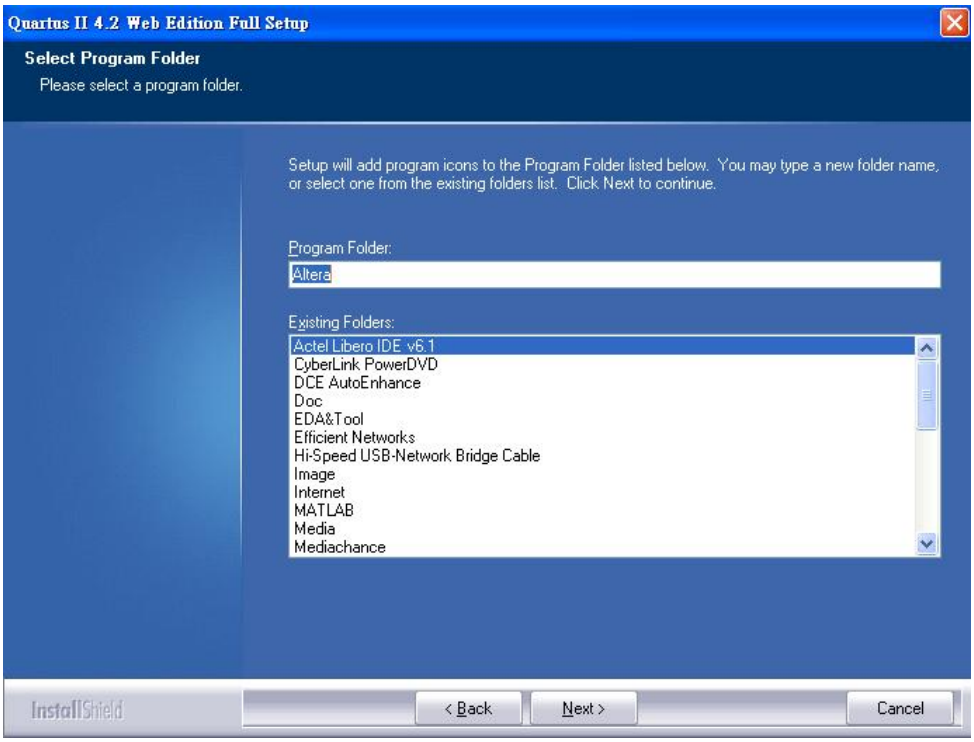
步驟5.　指定 qdesigns42 安裝路徑，按【Next】


Quartus II 4.2 Web Edition Full Setup
Choose Destination Location
Select folder where setup will install files.
Setup will install Quartus II 4.2 Web Edition Full Tutorial files in the following folder.
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.
Destination Folder
C:\altera\qdesigns42
Browse...
InstallShield
< Back
Next >
Cancel

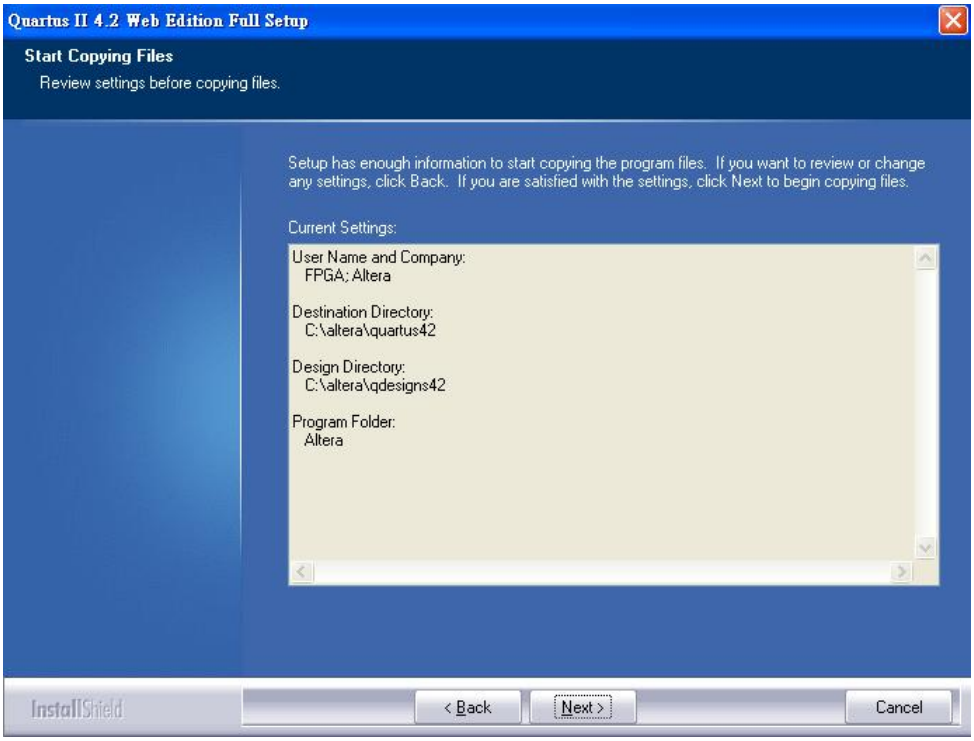
步驟6.　點選 Complete，按【Next】


Quartus II 4.2 Web Edition Full Setup
Setup Type
Select the setup type to install.
Please select a setup type.
Complete
All program features will be installed. (Requires the most disk space.)
Custom
Select which program features you want installed. Recommended for advanced users.
InstallShield
< Back
Next >
Cancel

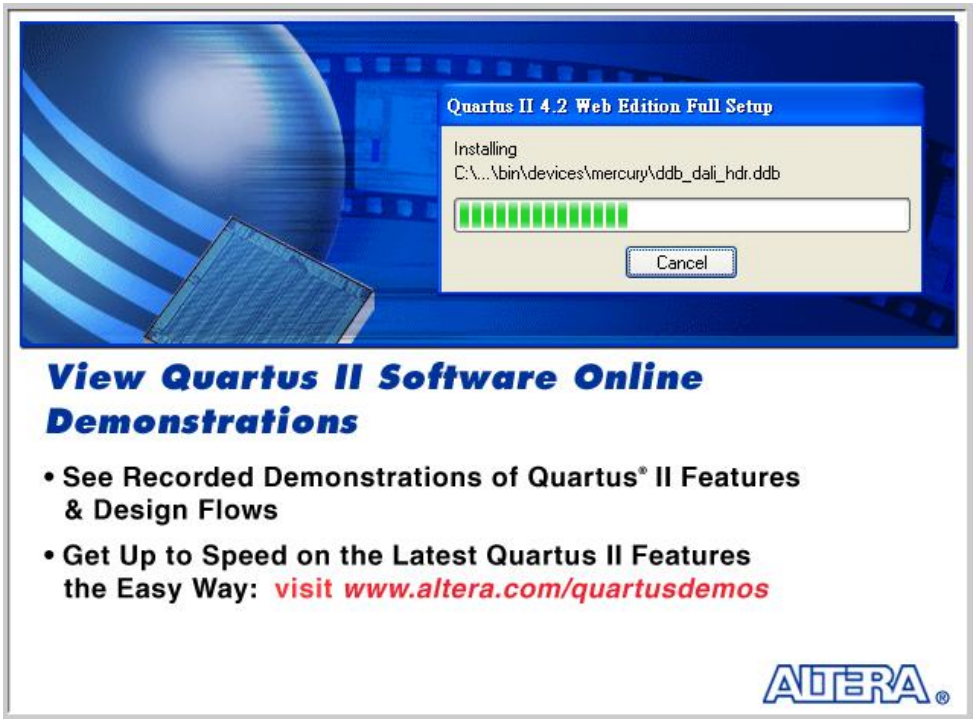
步驟7.　選擇程式資料夾，按【Next】



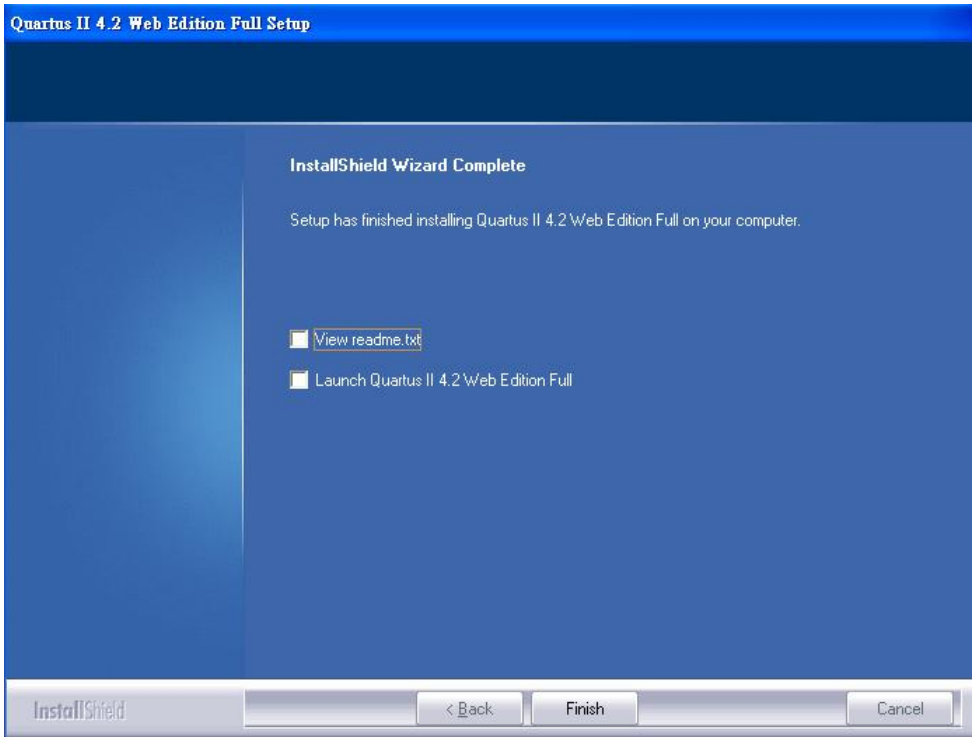
步驟8.　完成設定，按【Next】



步驟9.　開始進行安裝

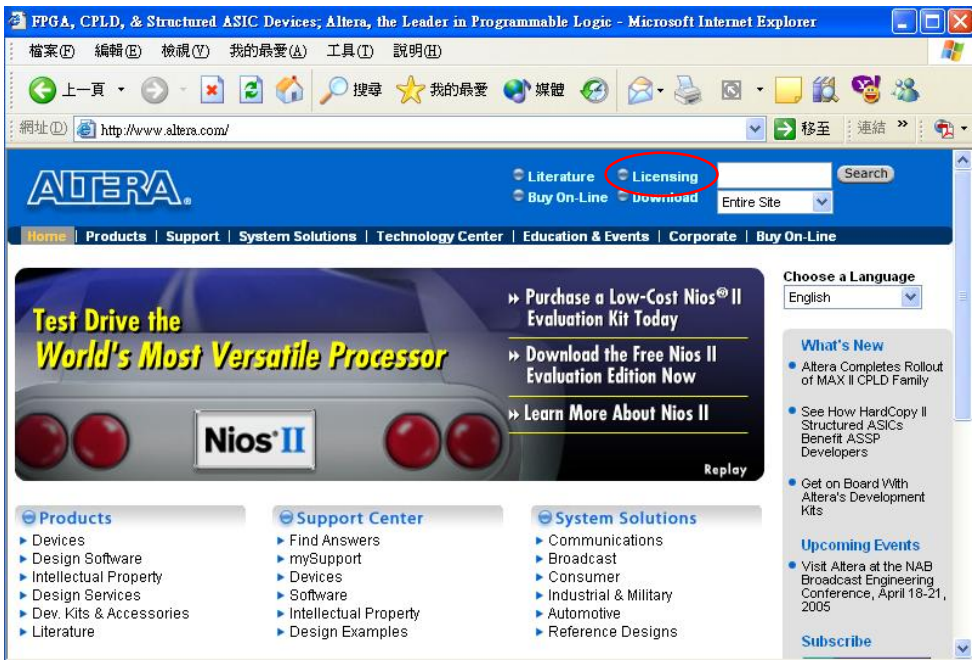


步驟10.　按【Finish】完成安裝

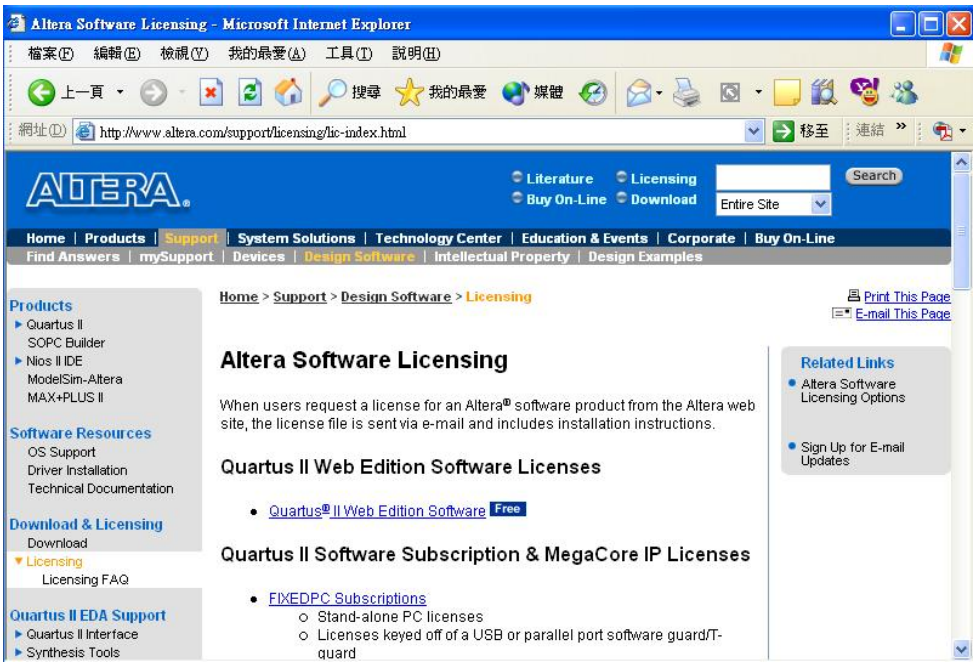


附錄 B　License 申請流程

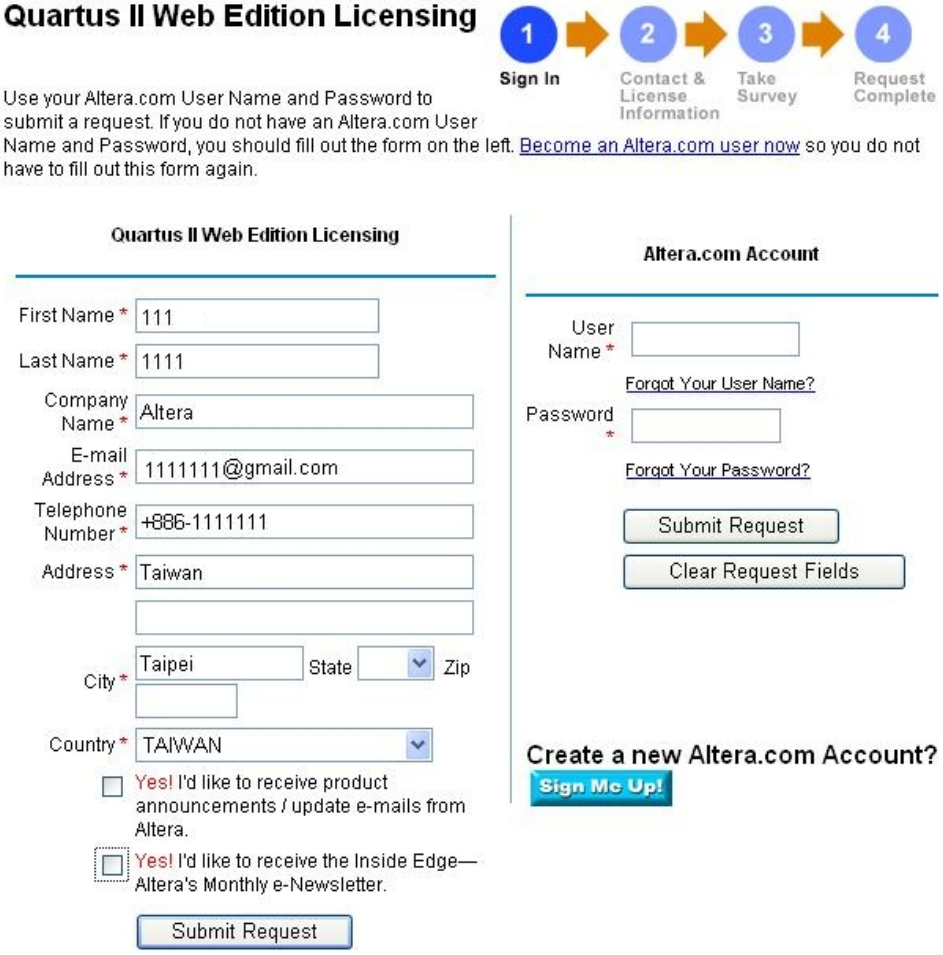
步驟1.　上網至 Altera 原廠網頁 http://www.altera.com，按【Licensing】



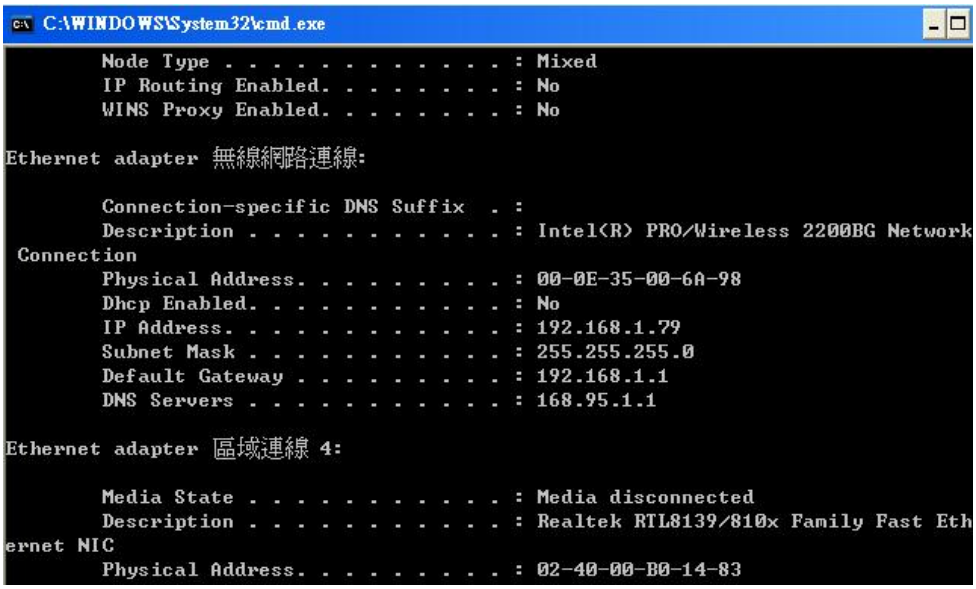
步驟2.　按 Quartus II Web Edition Software


Altera Software Licensing - Microsoft Internet Explorer
http://www.altera.com/support/licensing/lic-index.html
Literature
Licensing
Buy On-Line
Download
Search
Entire Site
Home | Products | Support | System Solutions | Technology Center | Education & Events | Corporate | Buy On-Line
Find Answers | mySupport | Devices | Design Software | Intellectual Property | Design Examples
Products
Quartus II
SOPC Builder
Nios II IDE
ModelSim-Altera
MAX+PLUS II
Software Resources
OS Support
Driver Installation
Technical Documentation
Download & Licensing
Download
Licensing
Licensing FAQ
Quartus II EDA Support
Quartus II Interface
Synthesis Tools
Home > Support > Design Software > Licensing
Print This Page
E-mail This Page
Altera Software Licensing
When users request a license for an Altera® software product from the Altera web site, the license file is sent via e-mail and includes installation instructions.
Quartus II Web Edition Software Licenses
Quartus® II Web Edition Software
Free
Quartus II Software Subscription & MegaCore IP Licenses
FIXEDPC Subscriptions
Stand-alone PC licenses
Licenses keyed off of a USB or parallel port software guard/T-guard
Related Links
Altera Software Licensing Options
Sign Up for E-mail Updates

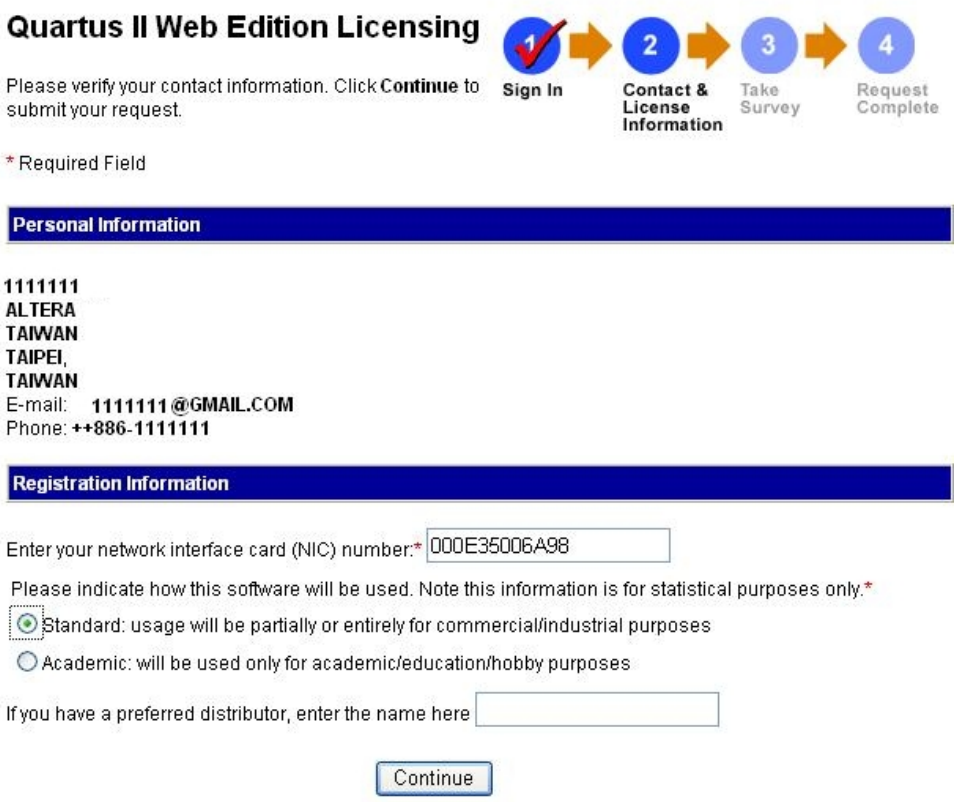
步驟3.　填寫個人資料，按【Submit Request】


Quartus II Web Edition Licensing
1 Sign In
2 Contact & License Information
3 Take Survey
4 Request Complete
Use your Altera.com User Name and Password to submit a request. If you do not have an Altera.com User Name and Password, you should fill out the form on the left. Become an Altera.com user now so you do not have to fill out this form again.
Quartus II Web Edition Licensing
First Name * 111
Last Name * 1111
Company Name * Altera
E-mail Address * 1111111@gmail.com
Telephone Number * +886-1111111
Address * Taiwan
City * Taipei
State
Zip
Country * TAIWAN
Yes! I'd like to receive product announcements / update e-mails from Altera.
Yes! I'd like to receive the Inside Edge—Altera's Monthly e-Newsletter.
Submit Request
Altera.com Account
User Name *
Forgot Your User Name?
Password *
Forgot Your Password?
Submit Request
Clear Request Fields
Create a new Altera.com Account?
Sign Me Up!

步驟4.　至【開始】【執行】鍵入 cmd，ipconfig －all ，記下電腦的網路卡序號 12 碼【Physical Address】


C:\WINDOWS\System32\cmd.exe
Node Type . . . . . : Mixed
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
Ethernet adapter 無線網路連線:
Connection-specific DNS Suffix . :
Description . . . . . : Intel(R) PRO/Wireless 2200BG Network Connection
Physical Address. . . . . : 00-0E-35-00-6A-98
Dhcp Enabled. . . . . : No
IP Address. . . . . : 192.168.1.79
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1
DNS Servers . . . . . : 168.95.1.1
Ethernet adapter 區域連線 4:
Media State . . . . . : Media disconnected
Description . . . . . : Realtek RTL8139/810x Family Fast Ethernet NIC
Physical Address. . . . . : 02-40-00-B0-14-83

步驟5.　輸入電腦的 12 碼網路卡序號，按【Continue】



Quartus II Web Edition Licensing

1 Sign In → 2 Contact & License Information → 3 Take Survey → 4 Request Complete

Please verify your contact information. Click **Continue** to submit your request.

* Required Field

Personal Information

1111111
ALTERA
TAIWAN
TAIPEI,
TAIWAN
E-mail: **1111111@GMAIL.COM**
Phone: **++886-1111111**

Registration Information

Enter your network interface card (NIC) number:* 000E35006A98

Please indicate how this software will be used. Note this information is for statistical purposes only.*

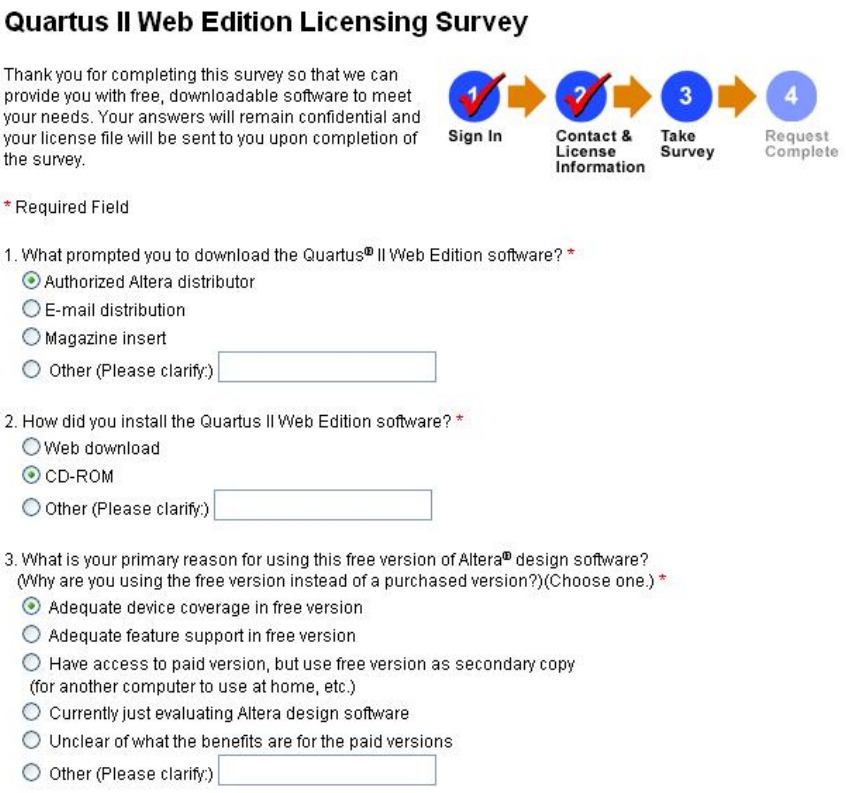
(•) Standard: usage will be partially or entirely for commercial/industrial purposes

( ) Academic: will be used only for academic/education/hobby purposes

If you have a preferred distributor, enter the name here

Continue

步驟6.　點選調查表，，按【Continue】



Quartus II Web Edition Licensing Survey

1 Sign In → 2 Contact & License Information → 3 Take Survey → 4 Request Complete

Thank you for completing this survey so that we can provide you with free, downloadable software to meet your needs. Your answers will remain confidential and your license file will be sent to you upon completion of the survey.

* Required Field

1. What prompted you to download the Quartus® II Web Edition software? *
   - (•) Authorized Altera distributor
   - ( ) E-mail distribution
   - ( ) Magazine insert
   - ( ) Other (Please clarify:)

2. How did you install the Quartus II Web Edition software? *
   - ( ) Web download
   - (•) CD-ROM
   - ( ) Other (Please clarify:)

3. What is your primary reason for using this free version of Altera® design software? (Why are you using the free version instead of a purchased version?)(Choose one.) *
   - (•) Adequate device coverage in free version
   - ( ) Adequate feature support in free version
   - ( ) Have access to paid version, but use free version as secondary copy (for another computer to use at home, etc.)
   - ( ) Currently just evaluating Altera design software
   - ( ) Unclear of what the benefits are for the paid versions
   - ( ) Other (Please clarify:)

步驟7.　完成 License 申請流程

Quartus II Web Edition Licensing Request Completed

Sign In → Contact & License Information → Take Survey → Request Complete

Thank you for choosing the Quartus II Web Edition development software. A license file enabling your free Quartus II Web Edition software will be e-mailed to you within 12 hours. Most license files are e-mailed to you within minutes. However, the delivery of the license file may be delayed between 1-12 hours in some cases. If you do not receive a license file immediately please wait at least 12 hours before requesting a second license.

Install Your License File

Instructions on how to install your new license file:

Installation Instructions for Quartus II Web Edition: Consolidated File
Installation Instructions for Quartus II Web Edition: Manual Files and Device Data Files

Introducing the Altera Subscription Program

After evaluating the Quartus II Web Edition development software, you may want to purchase the Altera Subscription Program, which provides Quartus II software and MAX+PLUS II software releases and updates for 12 months. Refer to the Altera Subscription Program page for complete details

步驟8.　至信箱中收信並下載 License 檔案儲存至電腦中

Your Altera Development Tools License Inbox

authorization@altera.com to me

Dear 1111111,
Thank you for requesting a license file to enable your Altera Quartus II software. Quartus II software delivers the highest average performance for FPGA and CPLD designs.

****

Quartus II Software - #1 in Performance

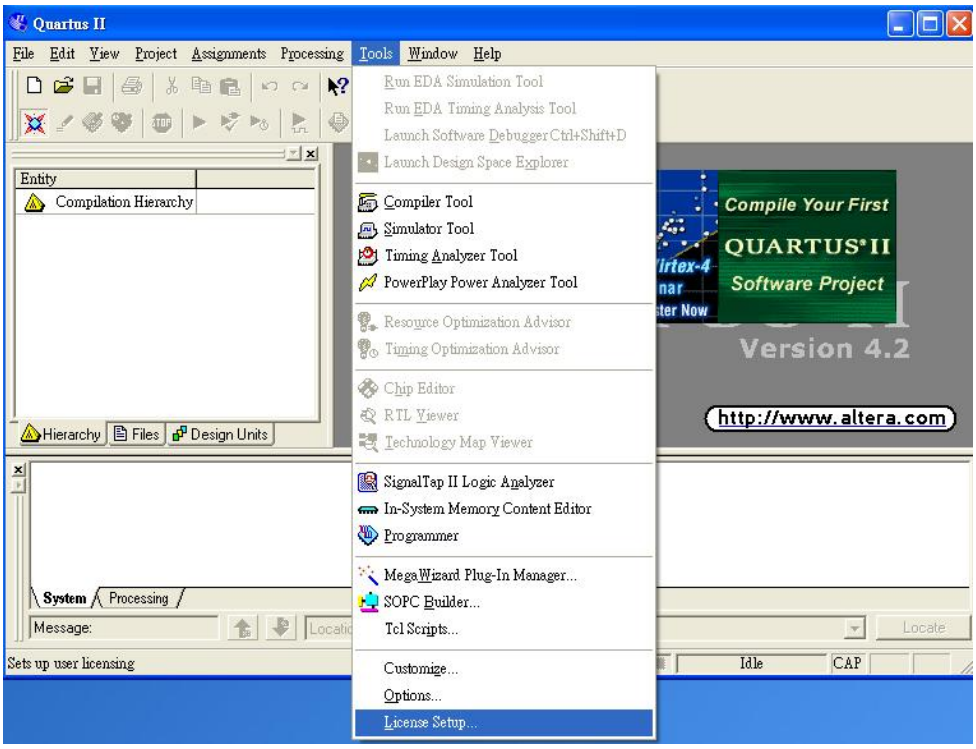
Details at www.altera.com/alterazone

****

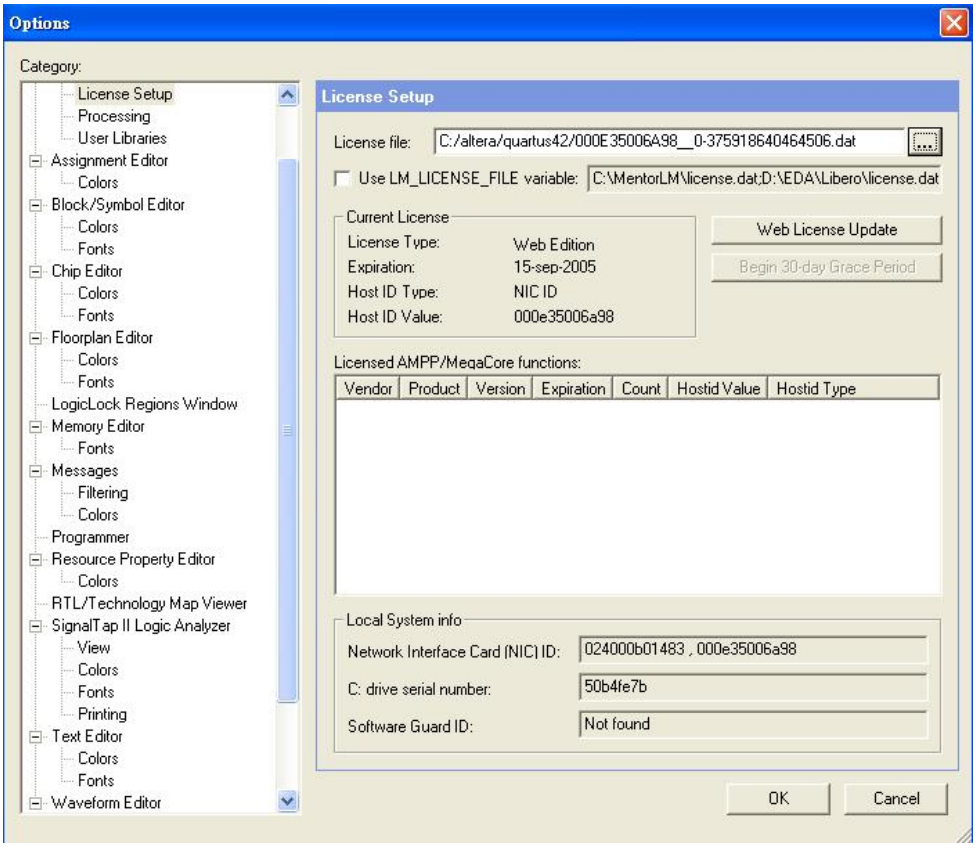
This email includes the following sections:

*Extensive Learning Resources
*License File Installation Instructions
*Support Information

步驟9.　點選【Tools】→【License Setup】



步驟10. 指定路徑至所下載的 License 檔

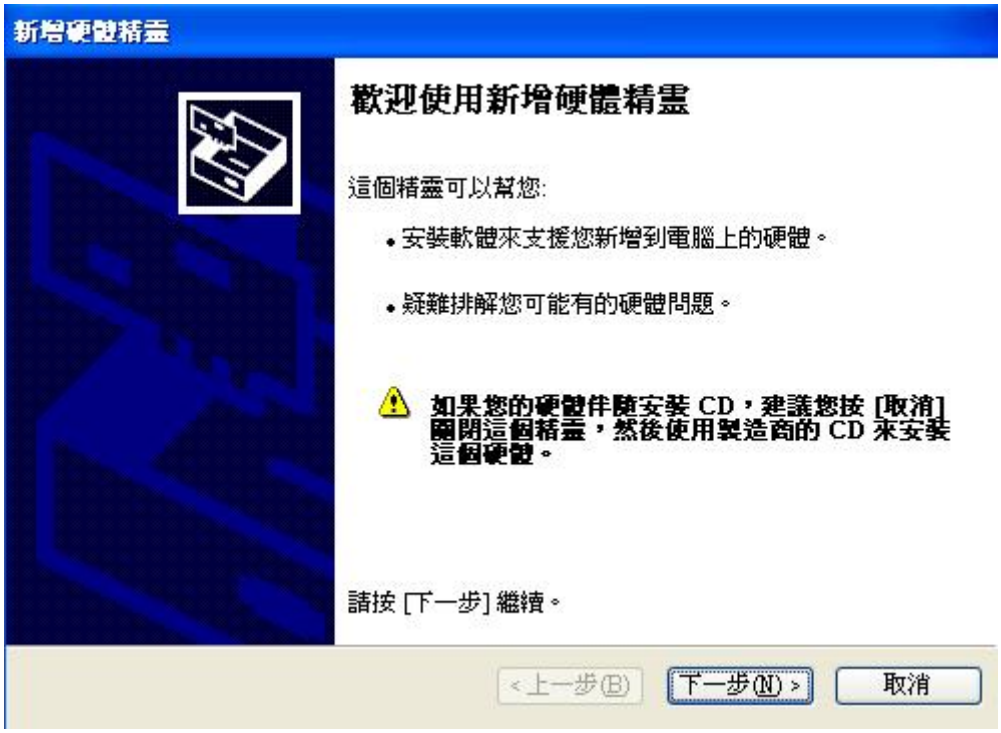


附錄 C　在 Windows XP/2000 下安裝 Download cable

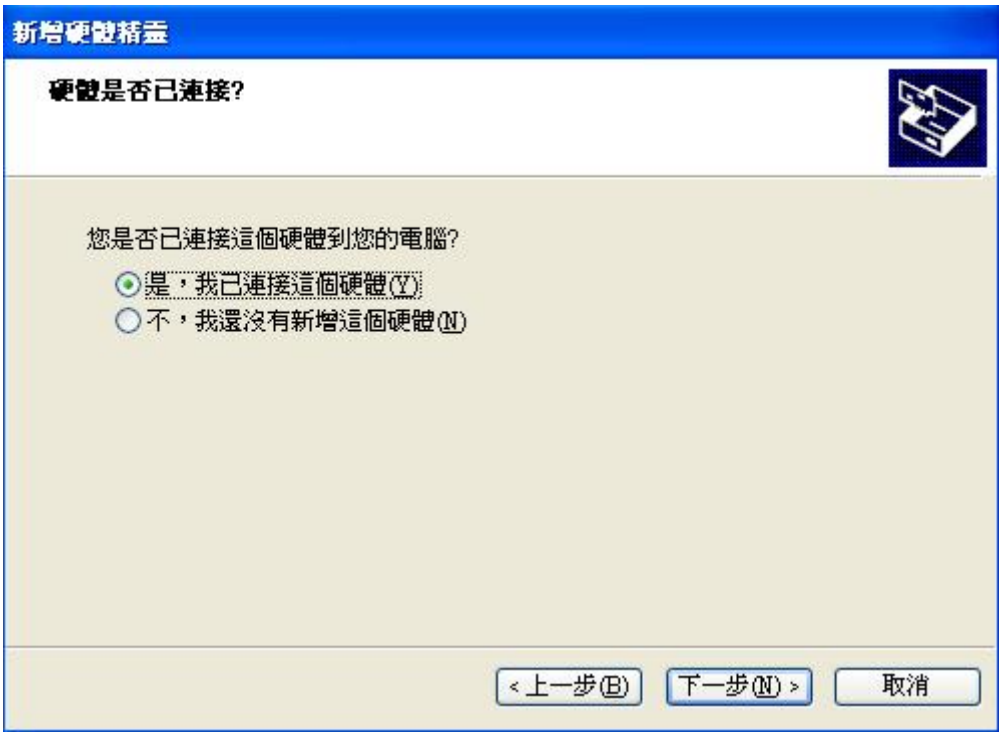
Download cable 是一個標準並列埠硬體介面，在 Windows 2000 與 XP 系統下需另外安裝驅動程式才能使用 LPT1 介面進行實驗板之晶片燒錄。

在 Windows XP 下新增硬體

步驟1.　【開始】→【設定】→【控制台】→【印表機和其他硬體】→【新增硬體】，出現新增硬體精靈，選取【下一步】。

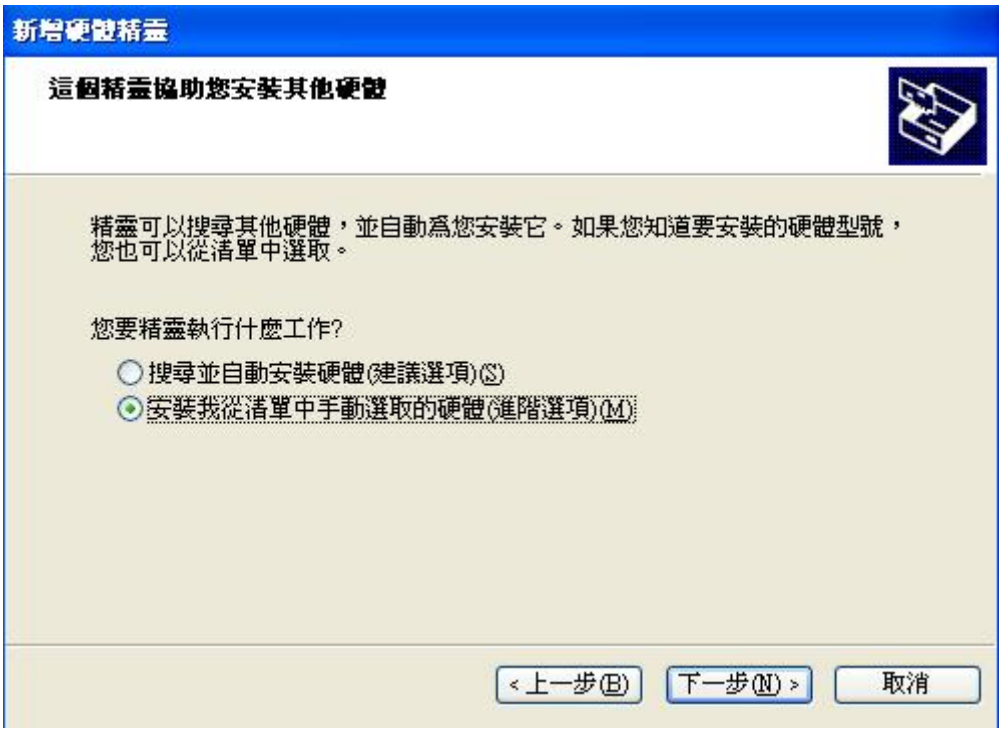


步驟2.　選擇【是，我已連接這個硬體】→【下一步】。



步驟3.　在新增硬體精靈視窗中選擇【新增硬體裝置】→【下一步】。

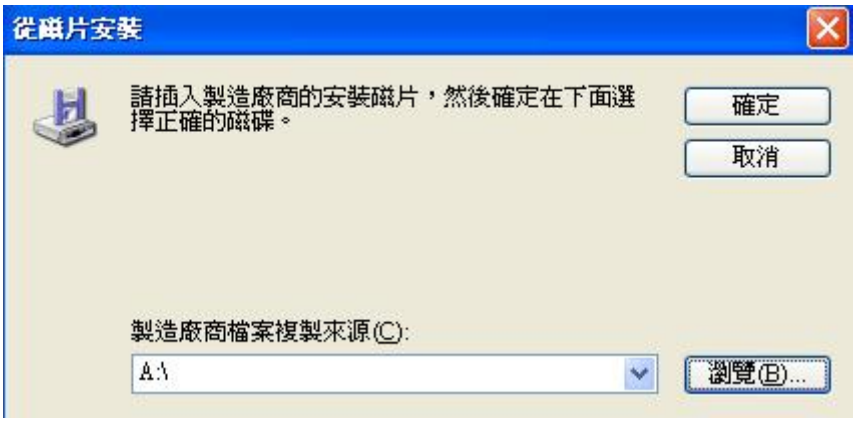
步驟4.　選擇【安裝我從清單中手動選取的硬體(進階選項)】→【下一步】。



步驟5.　選擇【音效，視訊及遊戲控制器】→【下一步】。

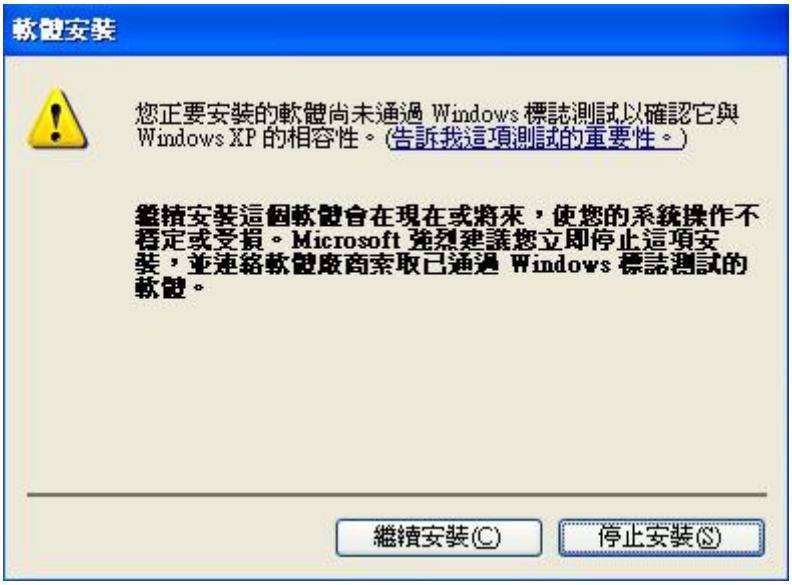
步驟6. 【從磁片安裝】→【下一步】。

步驟7. 按【瀏覽】，選擇驅動程式所在位置的資料夾。



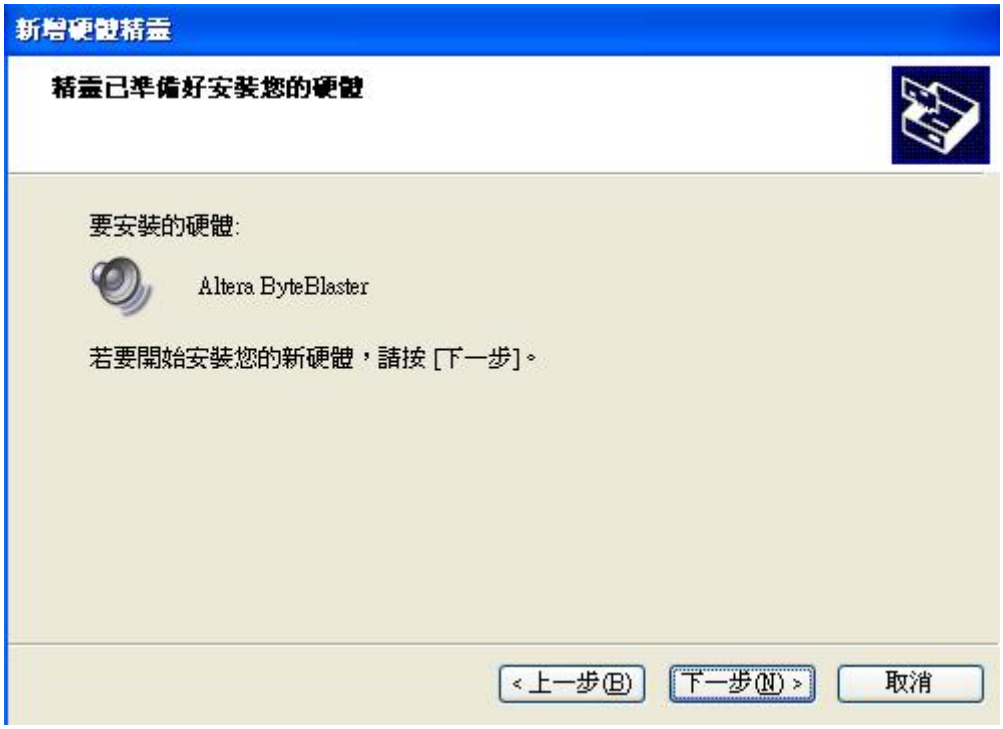
步驟8. 至『…\quartus42\drivers\win2000』目錄中找到 Win2000.inf 檔案後，選取【開啟】→【確定】

步驟9. 按下【繼續安裝】按鈕。

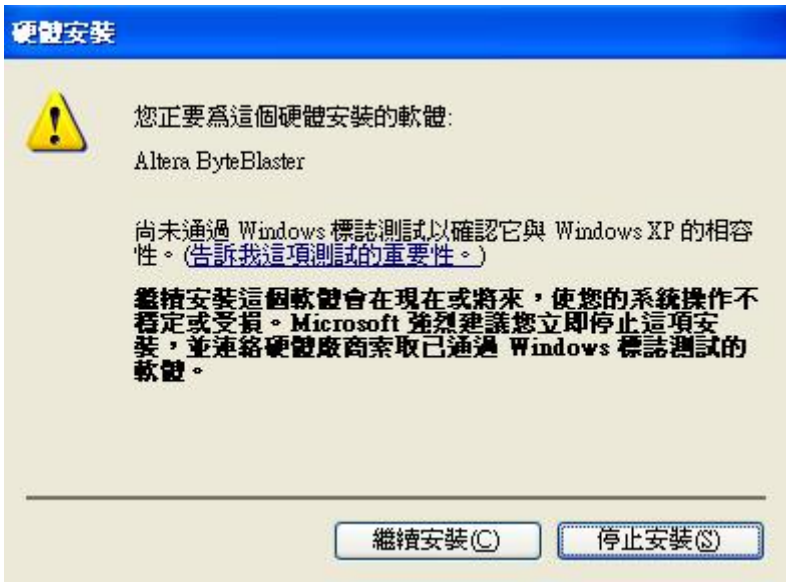


步驟10. 選取要為這個硬體安裝的裝置驅動程式：【Altera ByteBlaster】→【下一步】。

步驟11. 按【下一步】開始安裝新硬體。

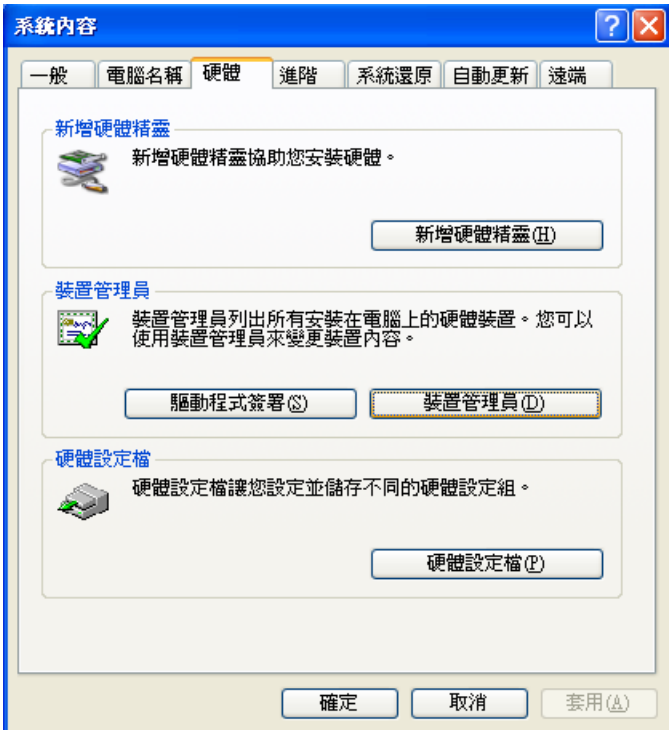


步驟12. 選擇【繼續安裝】。



步驟13. 按【完成】以關閉新增硬體精靈。

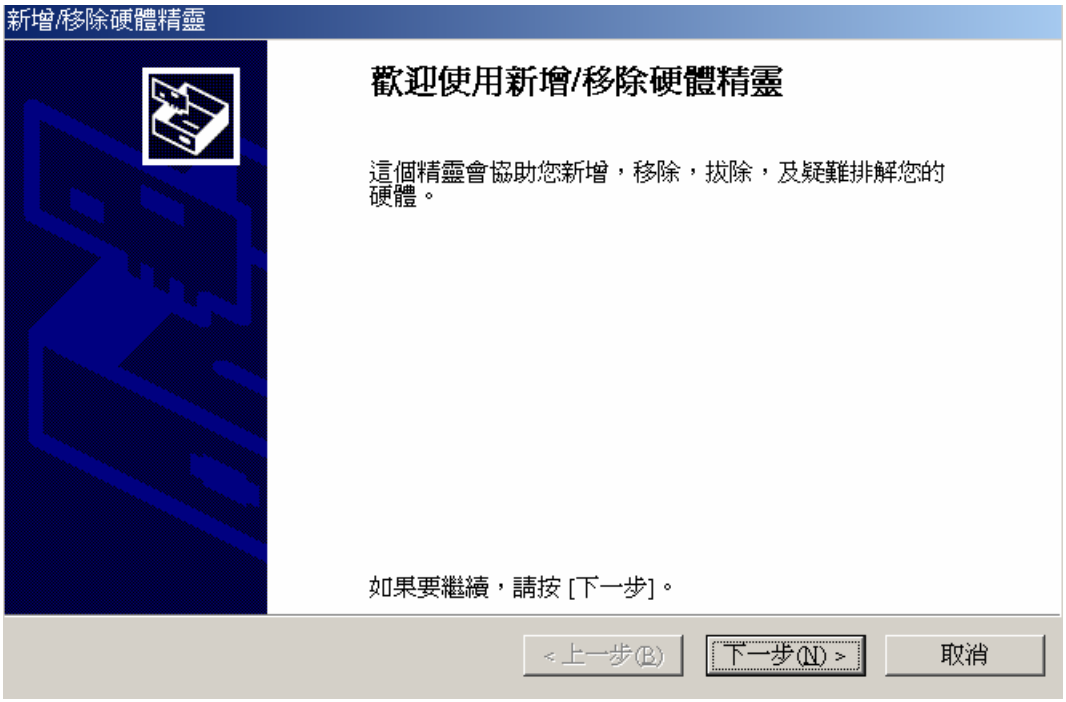
步驟14. 至【我的電腦】，按滑鼠右鍵選取【內容】，出現系統內容視窗，選取【硬體】下的【裝置管理員】。



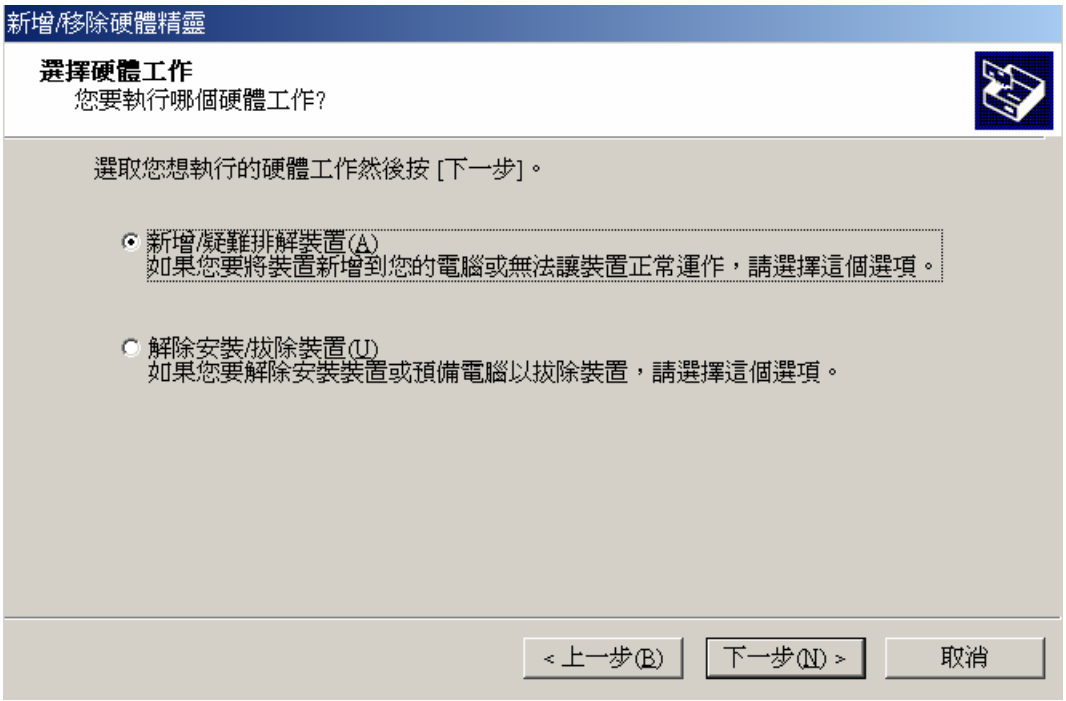
步驟15. 選取【音效，視訊及遊戲控制器】，若【Altera Byteblaster】出現在項目中，則表示安裝成功。

在 Windows2000 下新增硬體

步驟1. 選擇【開始】→【設定】→【控制台】→【新增/移除硬體】，出現新增移除硬體精靈，選取【下一步】。

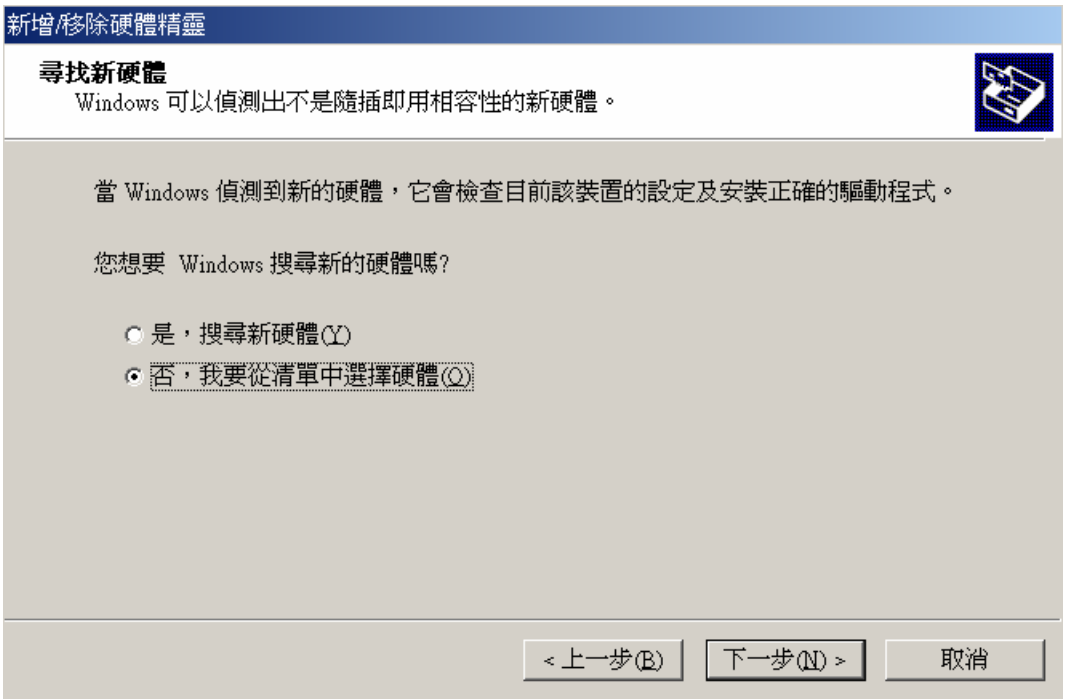


步驟2. 【新增/疑難排除裝置】→【下一步】。

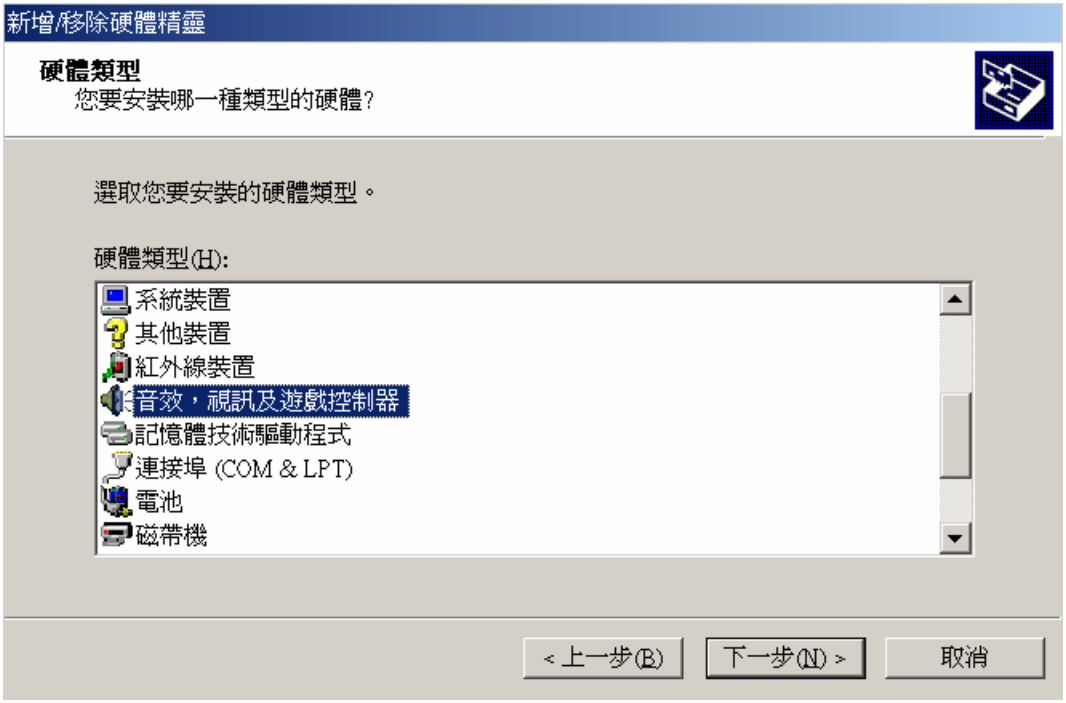


步驟3. 【新增一項裝置】→【下一步】。

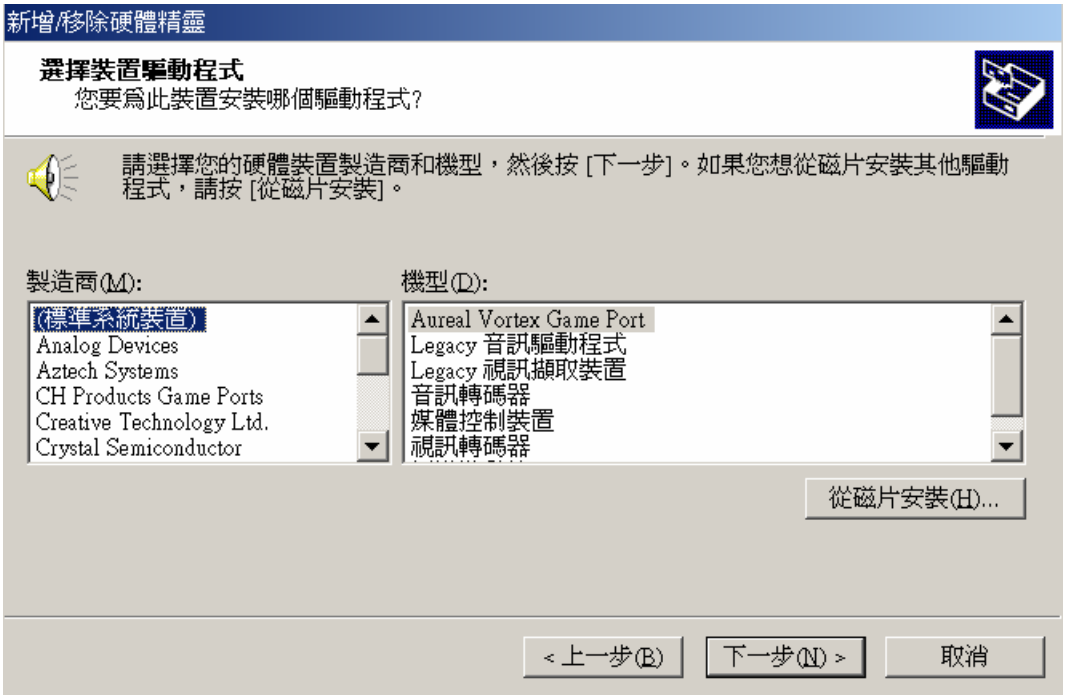
步驟4. 在新增/移除硬體精靈之操作視窗中，選擇【否，我要從清單中選取硬體】→【下一步】。



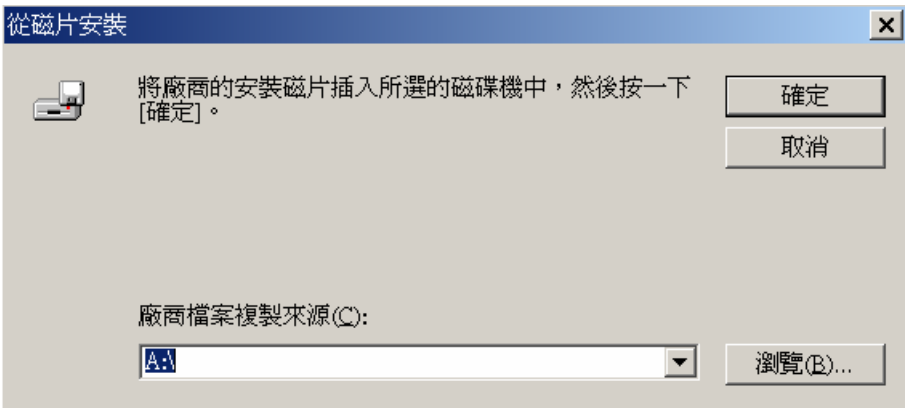
步驟5.　安裝【音效，視訊及遊戲控制器】→【下一步】。



步驟6.　【從磁片安裝】→【下一步】。



步驟7. 按【瀏覽】選項；至『…\quartus42\drivers\win2000』目錄中找到 win2000.inf 檔案後，選取【開啟】→【確定】。

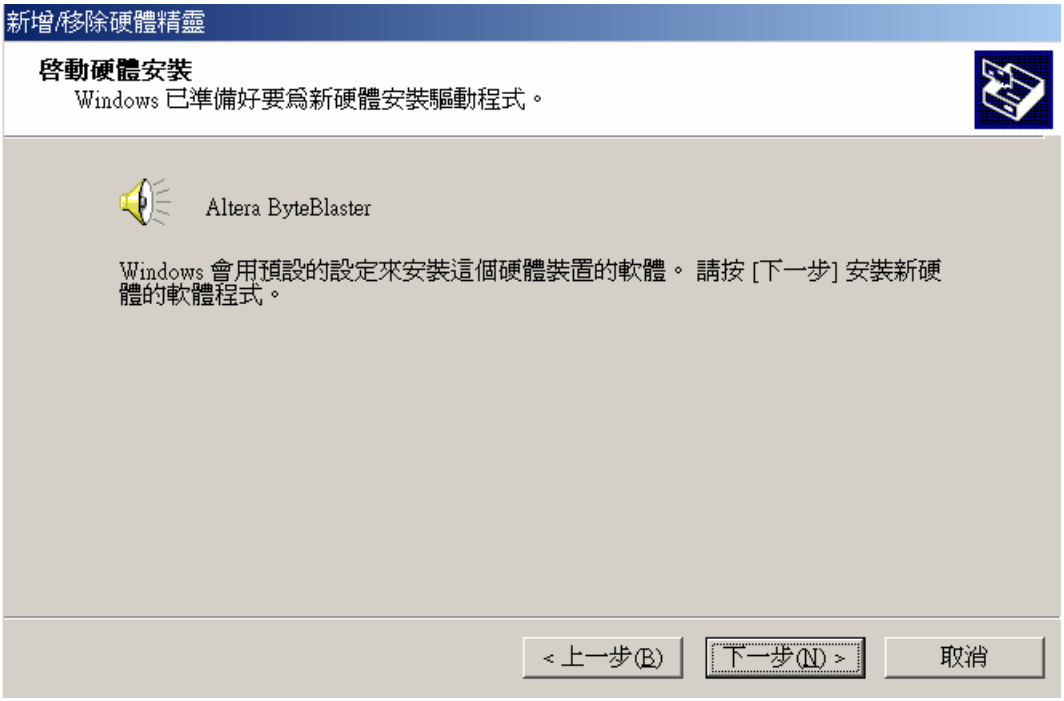


步驟8. 選擇【是】以繼續安裝作業。

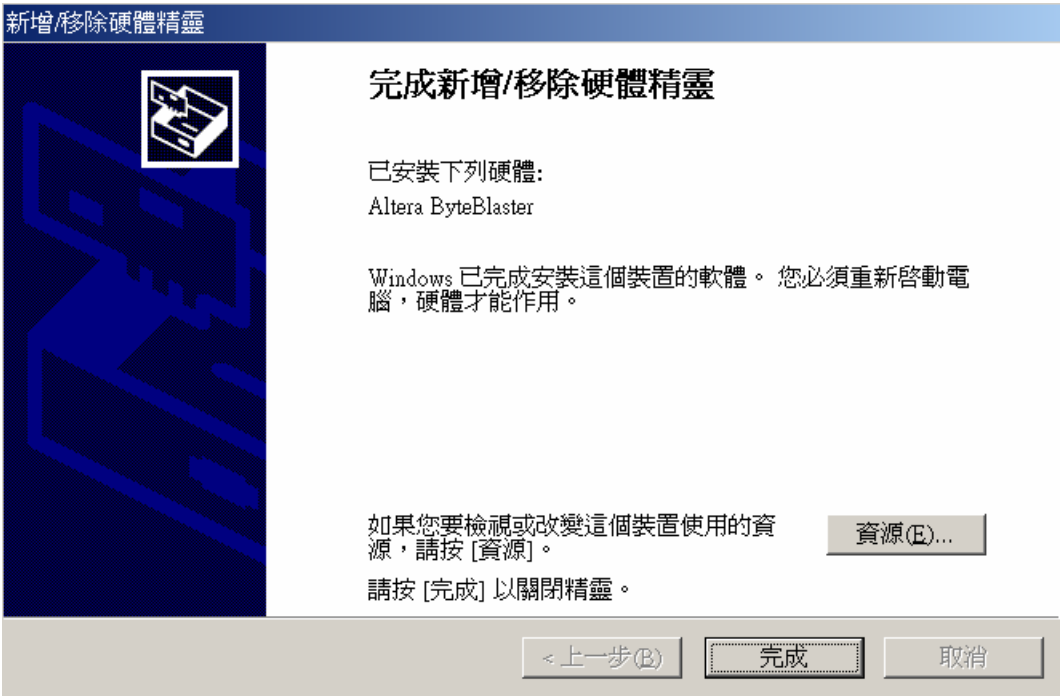
步驟9. 選擇【Altera ByteBlaster】→【下一步】。

步驟10. 選擇【是】以繼續安裝作業。

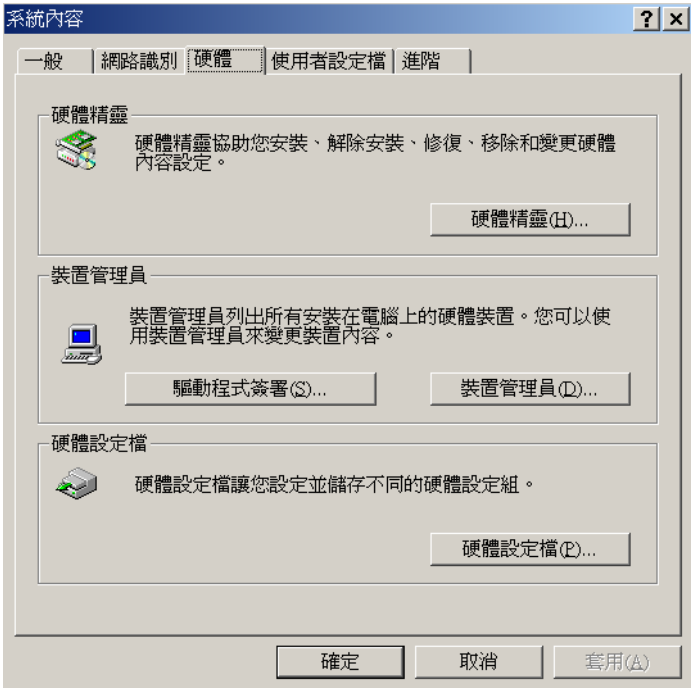
步驟11. 按【下一步】安裝新硬體的軟體程式。



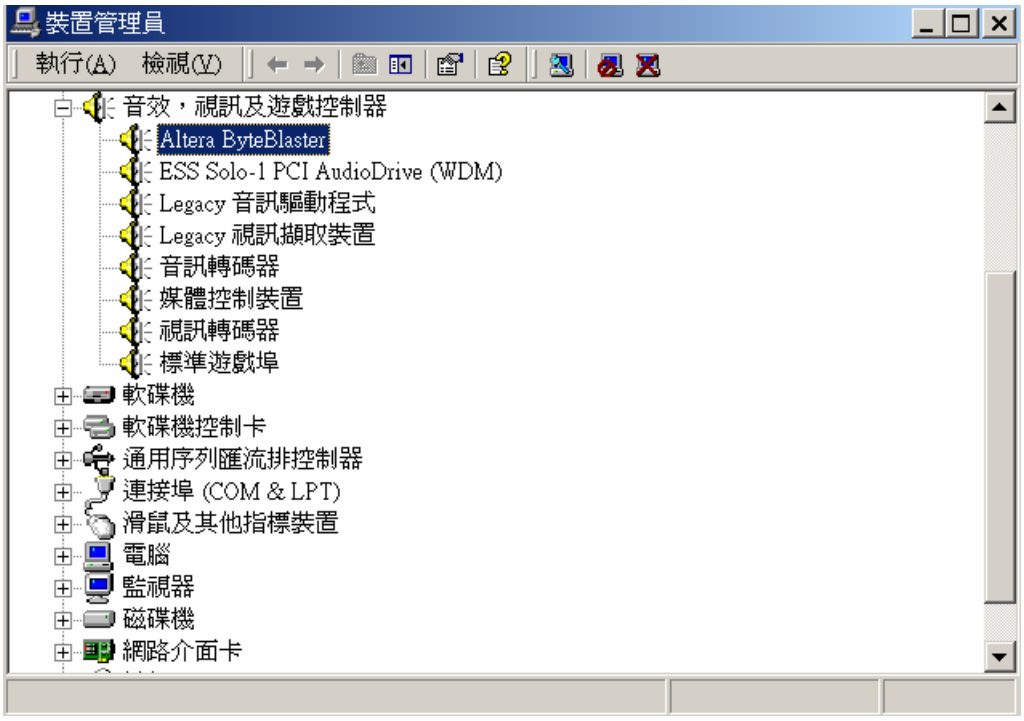
步驟12. 按【完成】以關閉新增/移除硬體精靈。



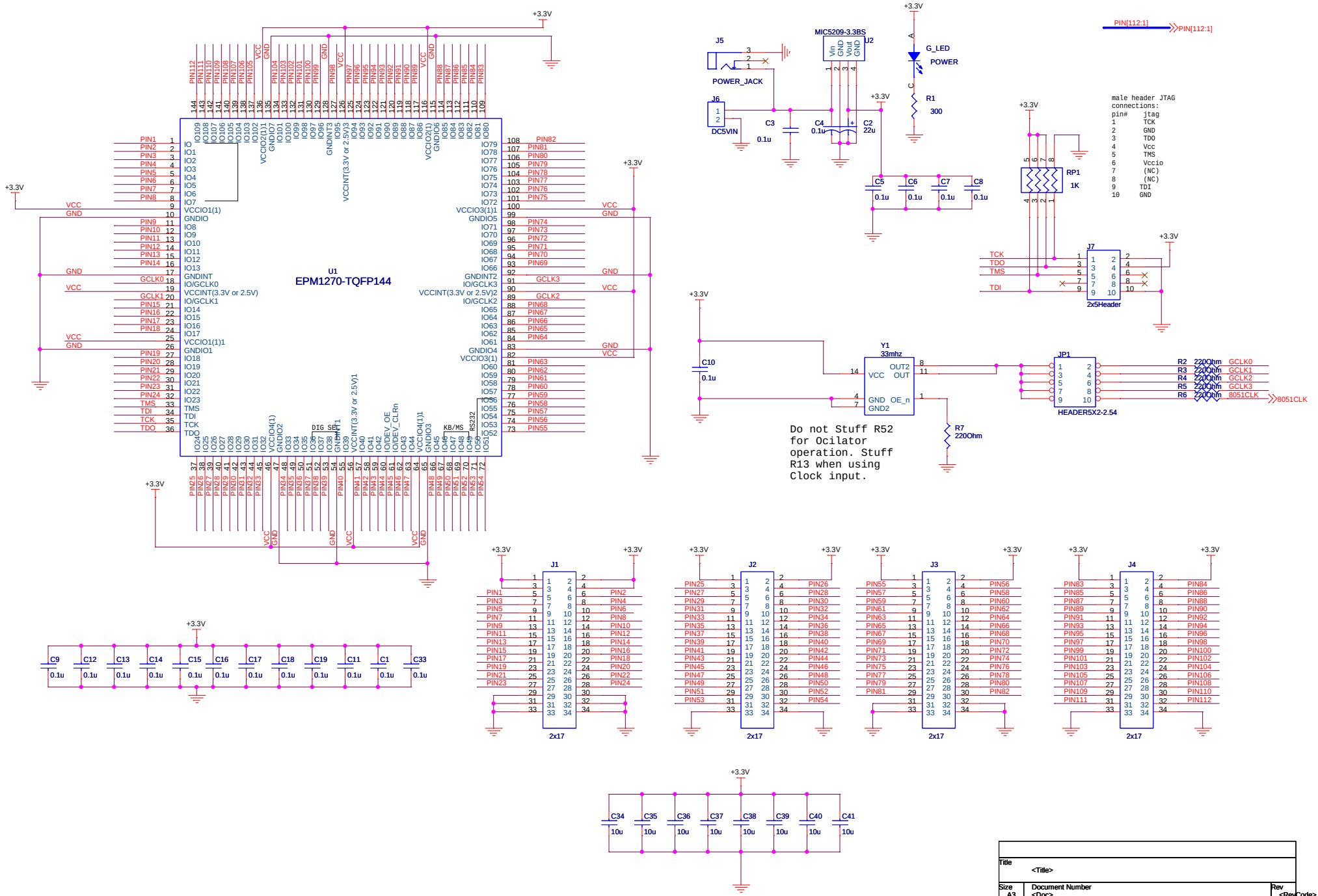
步驟13. 確定是否安裝成功，至桌面上選取【我的電腦】，按滑鼠右鍵選取【內容】，出現系統內容視窗，選取【硬體】下的【裝置管理員】。

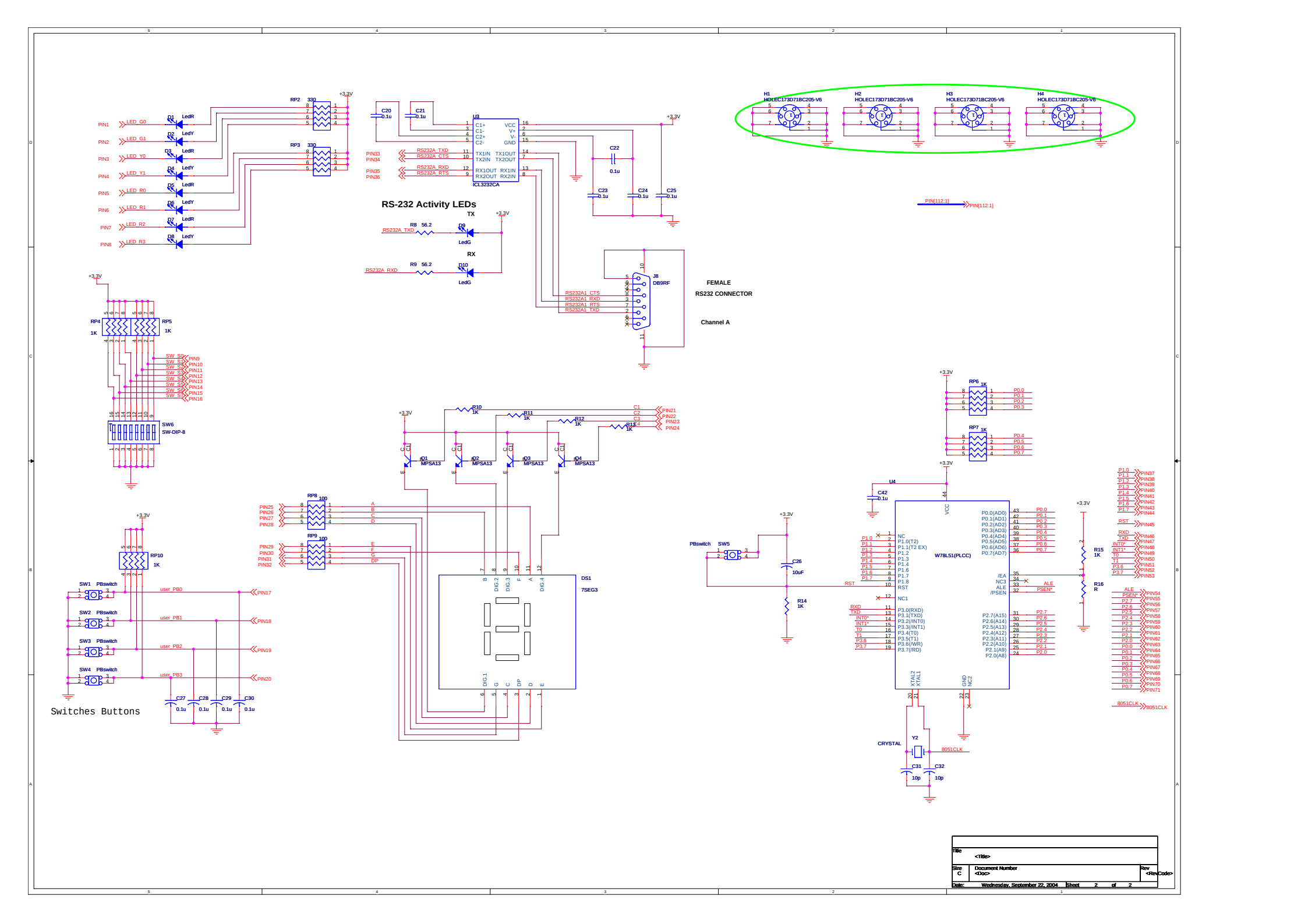


步驟14. 選取【音效，視訊及遊戲控制器】，若【Altera Byteblaster】出現在項目中，則表示安裝成功。



附錄 D　MAX II Starter Kit 電路圖


U1
EPM1270-TQFP144
J5
POWER_JACK
J6
DC5VIN
U2
MIC5209-3.3BS
G_LED
POWER
R1
300
C1
C2
22u
C3
C4
C5
C6
C7
C8
0.1u
RP1
1K
male header JTAG connections:
pin# jtag
1 TCK
2 GND
3 TDO
4 Vcc
5 TMS
6 Vccio
7 (NC)
8 (NC)
9 TDI
10 GND
J7
2x5Header
TCK
TDO
TMS
TDI
Y1
33mhz
C10
R7
220Ohm
JP1
HEADER5X2-2.54
R2 220Ohm GCLK0
R3 220Ohm GCLK1
R4 220Ohm GCLK2
R5 220Ohm GCLK3
R6 220Ohm 8051CLK
8051CLK
Do not Stuff R52 for Ocilator operation. Stuff R13 when using Clock input.
J1
J2
J3
J4
2x17
C9
C12
C13
C14
C15
C16
C17
C18
C19
C11
C33
C34
C35
C36
C37
C38
C39
C40
C41
10u
PIN[112:1]
+3.3V
VCC
GND
Title
<Title>
Size A3
Document Number <Doc>
Rev <RevCode>
Date: Wednesday, September 22, 2004
Sheet 1 of 2


RS-232 Activity LEDs
TX
RX
ICL3232CA
FEMALE
RS232 CONNECTOR
Channel A
SW-DIP-8
Switches Buttons
W78L51(PLCC)
7SEG3
CRYSTAL
H1 HOLEC173D71BC205-V6
H2 HOLEC173D71BC205-V6
H3 HOLEC173D71BC205-V6
H4 HOLEC173D71BC205-V6
Title
<Title>
Size C
Document Number
<Doc>
Rev
<RevCode>
Date: Wednesday, September 22, 2004
Sheet 2 of 2